PROCEEDINGS
SEKE 2023

The 35th International Conference on
Software Engineering &
Knowledge Engineering

Sponsored by
KSI Research Inc. and Knowledge Systems Institute, USA

Technical Program
July 1 – 10, 2023

Larkspur Landing South San Francisco Hotel, USA and
KSIR Virtual Conference Center, USA

Organized by
KSI Research Inc. and Knowledge Systems Institute, USA
SEKE 2023
The 35th International Conference on
Software Engineering &
Knowledge Engineering

July 1 – 10, 2023
Larkspur Landing South San Francisco Hotel, USA and
KSIR Virtual Conference Center, USA

Conference Organization

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

PROGRAM COMMITTEE CHAIR AND CO-CHAIR
Giuseppe Polese, University of Salerno, Italy; Program Chair
Kamran Sartipi, East Carolina University, USA; Program Co-Chair
Iaakov Exman, Holon Institute of Technology, Holon, Israel; Program 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, Univ. of Connecticut, USA
Natalia Juristo, Universidad Politecnica de Madrid, Spain
Taghi Khoshgoftaar, Florida Atlantic University, USA
PROGRAM COMMITTEE

Silvia Acuna, Universidad Autonoma de Madrid, Spain
Shibbir Ahmed, Iowa State University, USA
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
Fevzi Belli, University of Paderborn, Germany
Ateet Bhalla, Consultant, India
Swapan Bhattacharya, Jadavpur University, India
Tannay Bhowmik, Mississippi State University, USA
Michael Bosu, Waikato Institute of Technology, New Zealand
Ivo Bukovsky, Czech Technical University in Prague, Czech Republic
Guoray Cai, Penn State University, USA
Rafael Cardoso, University of Aberdeen, United Kingdom
John Castro, Universidad de Atacama, Chile
Lingwei Chen, Pennsylvania State University, USA
Wen-Hui Chen, NationalTaipei University of Technology, Taiwan
Xiang Chen, Nantong University, China
Xiangping Chen, Sun Yat-sen 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
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, Holon Institute of Technology, Holon, Israel
Onyeka Ezenwoye, Augusta University, USA
Yuan Fei, Shanghai Normal University, China
Maria Francesca Costabile, University of Bari, Italy
Jianbo Gao, Peking University, China
Honghao Gao, ShangHai University, China
Raul Garcia-Castro, Universidad Politécnica 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
Antonio Gutierrez, Johannes Kepler University, Austria
Hao Han, National Institute of Informatics, Japan
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 Mexico Tech, USA
Shuyuan Jin, Sun Yat-sen University, China
Peiquan Jin, University of Science and Technology, China
Jason Jung, Chung-Ang University, South Korea
Pankaj Kamthan, Concordia University, Canada
Taeghyun Kang, University of Central Missouri, USA
Wiem Khlif, Miracle Laboratory, Tunisia
Taghi Khoshgoftaar, Florida Atlantic University, USA
Jun Kong, North Dakota State University, USA
Luyao Kou, Tsinghua University, China
Vinay Kulkarni, Tata Consultancy Services, India
Samantha Kumara, Sabaragamuwa University of Sri Lanka, Sri Lanka
Olivier LeGoaer, University of Pau, France
Meira Levy, Shenkar College of Engineering and Design, Israel
Peng Liang, Wuhan University, China
Tong Li, Beijing University of Technology, China
Xin Li, Google Inc., USA
Yanhui Li, Nanjing University, China
Yingling Li, Chengdu University of Information Technology, China
Zengyang Li, Central China Normal University, China
Zhi Li, Guangxi Normal University, China
Lan Lin, Ball State University, USA
Bixin Li, Southeast University, China
Xiaodong Liu, Edinburgh Napier University, United Kingdom
Weidong Liu, Inner Mongolia University, China
Wanwei Liu, National University of Defense Technology, China
Yi Liu, University of Massachusetts Dartmouth, USA
Zhijie Liu, ShanghaiTech University, China
Jiawei Lu, Zhejiang University of Technology, China
Xinjun Mao, National University of Defense Technology, China
Kristof Meixner, TU Wien, Austria
Andre Menolli, Universidade Estadual do Norte do Parana (UENP), Brazil
Hind Milhem, University of Ottawa, Canada
Ran Mo, Central China Normal University, China
Hiroyuki Nakagawa, Osaka University, Japan
Masaki Nakamura, Toyama Prefectural University, Japan
Sumana Nath, University of Saskatchewan, Canada
Alex Norta, Tallinn University, Estonia
Kazuhiro Ogata, JAIST, Japan
Shinpei Ogata, Shinshu University, Japan
Carlos Pantoja, Federal Center for Technological Education (CEFET-RJ), Brazil
George Papadopoulos, University of Cyprus, Cyprus
Hyungbae Park, University of Central Missouri, USA
David Parsons, The Mind Lab, New Zealand (SEKEE)
Rong Peng, Wuhan University, China
Angelo Perkusich, Federal University of Campina Grande, Brazil
Giuseppe Polese, University of Salerno, Italy
Chen Qian, Donghua University in Shanghai, China
Rick Rabiser, Johannes Kepler University, Austria
Claudia Raibulet, University of Milan, Italy
Damith Rajapakse, National University of Singapore, Singapore
Rajeev Raje, IUPUI, USA
Marek Reformat, University of Alberta, Canada
Daniel Rodriguez, Universidad de Alcala, Spain
Claudio Sant'Anna, Universidade Federal da Bahia, Brazil
Kamran Sartipi, East Carolina University, USA
Nazlie Shahmir, Canadian Pacific Railway Limited, Canada
Kaize Shi, Beijing Institute of Technology, China
Michael Shin, Texas Tech University, USA
Saeed Siddik, University of Dhaka, Bangladesh
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 Touré, 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 Vaz Alves, Federal University of Technology - Parana (UTFPR), Brazil
Gennaro Vessio, University of Bari, Italy
José Viterbo, Fluminense Federal University (UFF), Brazil
Chenyang Wang, Tsinghua University, China
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
Xinzhi Wang, Shanghai University, China
Ye Wang, Zhejiang Gongshang University, China
Lingwei Wei, Chinese Academy of Sciences, China
Michael Weiss, Carleton University, Canada
Franz Wotawa, TU Graz, Austria
Ji Wu, Beihang University, 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, Japan
Guowei Yang, Texas State University, USA
Dongjin Yu, Hangzhou Dianzi University, China
Huiqun Yu, East China University of Science and Technology, China
Jiang Yue, Fujian Normal University, China
Fiorella Zampetti, University of Sannio, Italy
Du Zhang, Macau University of Science and Technology, China
Pengcheng Zhang, Hohai University, China
Xiaofang Zhang, Soochow University, China
BEST DEMO AWARD COMMITTEE
Kuhaneswaran Banujan, Sabaragamuwa University of Sri Lanka; Chair, Best Demo Award, Sri Lanka

PUBLICITY CHAIR AND CO-CHAIR
Patrick Cook, Texas Tech University, USA; Publicity 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 35th International Conference on Software Engineering and Knowledge Engineering (SEKE). 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, knowledge engineering, and their reciprocal influences. Indeed, knowledge engineering has historically contributed to the development of software engineering through knowledge-based automated tools, and nowadays machine learning is widely used to perform several empirical activities of software engineering. On the other hand, the recent boom of artificial intelligence calls for standard development processes for modern intelligent systems. In this evolving scenario, 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 an unprecedented 218 submissions covering main areas of software engineering and artificial intelligence, including papers at the intersection between these two disciplines, with particular emphasis, as expected, on machine learning approaches for software engineering activities. 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 84 full papers for the general conference (39 percent), 61 short papers (28 percent) and two poster papers. SEKE 2023 is the first SEKE conference consisting of both a Live Conference in Larkspur Landing South San Francisco Hotel, USA and a Virtual Conference in KSIIR Virtual Conference Center, USA to suit the different needs of conference authors and attendees. The SEKE Live Conference consists of one keynote session, two panel sessions, four tutorials, two invited sessions and three paper presentation sessions. The SEKE 2023 Virtual Conference consists of nine paper presentation sessions. 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 2023 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 Dr. Shi-Kuo Chang, Founder and Steering Committee Chair of the SEKE Conference Series for his guidance. Then we like to thank 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 2023.

Giuseppe Polese, University of Salerno, Italy; Program Chair
Kamran Sartipi, East Carolina University, USA; Program Co-Chair
Iaakov Exman, Holon Institute of Technology, Holon, Israel; Program Co-Chair
Keynote

Quantum Software Engineering: Practical Challenges

Professor Mario Piattini
University of Castilla-La Mancha, Spain

Abstract

If we consider the 19th century as the machine age, and the 20th century as the information age, the 21st century will be the quantum age. In fact, since the beginning of the "second quantum revolution" in the 1980s, several algorithms and quantum computers based on different technologies have been proposed and have been able to demonstrate their great advantage over "classical" computers. There are already numerous quantum programming languages, quantum software development environments and platforms. In this talk we will give an overview of the state of the art and practice in relation to Quantum Software Engineering & Programming, and we will look at some of the challenges posed by the practical application of quantum software development.

About the Speaker

Prof. Mario Piattini. PhD in Computer Science from UPM (Polytechnic University of Madrid, Spain). He has certificates (CISA, CISM, CRISC and CGEIT) by ISACA (in Illinois, USA), and from PMP (Project Management Professional). He is a Full Professor at the Escuela Superior de Informatica of the University of Castilla-La Mancha, Spain, where he directs the Alarcos Research Group, specialized in Information Systems Quality. He is also a Quantum Chief Research Officer, promoter of The Talavera Manifesto for Quantum Software Engineering and Programming, and coeditor of the book "Quantum Software Engineering", published by Springer, in 2022.
Panel Session WSE: Women in Software Engineering

Moderator: Pankaj Kamthan, Concordia University, Canada

Panelists:

Edna Canedo, University of Brasília, Brasília, Brazil

Mirella Moro, Federal University of Minas Gerais, Belo Horizonte, Brazil

Nazlie Shahmir, Canadian Pacific Railway Limited, Calgary, Canada

Bianca Trinkenreich, Northern Arizona University, Flagstaff, USA
Panel Session QSE: Quantum Software Engineering

**Moderator:** Prof. Iaakov Exman, Holon Institute of Technology, Israel

**Panelists:**

Prof. Rui Maranhao

Prof. Mario Piattini, University of Castilla-La Mancha, Spain
# Table of Contents

## Session SCE: Social Computing and Engineering

*Xinzhi Wang, Mengyue Li, Yige Yao, Zhennan Li, Yi Liu and Hui Zhang*

**Short Review of Intention Mining in Social Crisis Management through Automatic Technologies**  

*Qitao Hu and Xiao Wei*

**Topic and Speaker-aware Hierarchical Encoder-Decoder Model for Dialogue Generation**  

*Senjun Pei and Weidong Liu*

**Convolution Neural Network Based Patent Infringement Detection Method**  

*Wenwen Xiao, Xinzhi Wang, Xiangfeng Luo and Shaorong Xie*

**DDCL: A Dual Decision-making Continuous Reinforcement Learning Method Based on Sim2Real**  

*Fayrouz Elsalmy, Nada Hany Sherief and Walid Abdelmoez*

**Inclusive Gamification: An Exploratory Study in Software Development Enterprises (S)**  

## Session SEKE1: Software Engineering and Knowledge Engineering

*Ryo Soga, Hideyuki Kanuka, Takatomi Kubo, Takashi Ishio and Kenichi Matsumoto*

**Will you use software development support using biosignals? A survey from software developers**  

*Melody Hammel and Lan Lin*

**Assuring Domain Software Quality through Workflow Testing and Specification**  

*Ju Ma, Yi-Yang Sun, Peng He and Zhang-Fan Zeng*

**GSAGE2defect: An Improved Approach to Software Defect Prediction based on Inductive Graph Neural Network**  

*Pankaj Kamthan and Nazlie Shahmir*

**Small Educational Steps Towards Improving the Status of Women in Software**
Engineering

Yuanchi Ma, Hui He and Zhendong Niu
BDC: Using BERT and Deep Clustering to Improve Chinese Proper Noun Recognition

Session SEKE2: Software Engineering and Knowledge Engineering

Keita Murase and Shingo Takada
Applying Reinforcement Learning for Automated Testing of Mobile Application Focusing on State Definition, Reward, and Learning Method

Carlos Pantoja, João Victor Guinelli, Tielle Alexandre and Jose Viterbo
A Model-Driven Development Framework for Geographical and Relational Database Systems

Masaya Fujimoto, Hiroyuki Nakagawa and Tatsuhiro Tsuchiya
Expansion Mechanism for Runtime Verification of Self-adaptive Systems

Jing Zhou, Shan Leng, Fang Wang and Hansheng Wang
Automatic Discovery of Controversial Legal Judgments by an Entropy-Based Measurement (S)

Wasja Brunotte, Jakob Droste and Kurt Schneider
Context, Content, Consent - How to Design User-Centered Privacy Explanations (S)

Session TSECQS: Theoretical Software Engineering, Classical and Quantum Software Similarities

Iaakov Exman and Ariel Zvulunov
Quantum Software Models: Quantum Modules Tomography and Recovery Theorem

Krishn Vishwas Kher, Ishan Joshi, Bharat Chandra Mukkavalli, Lei Zhang, M. V. Panduranga Rao
Automatic Diagnosis of Quantum Software Bug Fix Motifs

Canh Minh Do and Kazuhiro Ogata
Symbolic Model Checking Quantum Circuits in Maude
Session SEKE3: Software Engineering and Knowledge Engineering

Yihao Zhang, Zeming Wei, Xiyue Zhang and Meng Sun
Using Z3 for Formal Modeling and Verification of FNN Global Robustness (S) 110

Marco Fiore, Marina Mongiello, Giovanni Tricarico, Francesco Bozzo, Cinzia Montemurro, Alessandro Petrontino, Clemente Giambattista and Giorgio Mercuri
Blockchain-based Food Traceability System for Apulian Marketplace: Enhancing Transparency and Accountability in the Food Supply Chain (S) 114

Kazusa Takemoto and Shingo Takada
Applying Symbolic Execution to Semantic Code Clone Detection (S) 118

Session ESSE: Empirical Studies and Software Engineering

Beiqi Zhang, Peng Liang, Xiyu Zhou, Aakash Ahmad and Muhammad Waseem
Practices and Challenges of Using GitHub Copilot: An Empirical Study 124

Qing Mi, Haotian Bai, Xiaozhou Wang, Wenrui Liu and Xingyue Song
An Empirical Study of Coding Style Compliance on Stack Overflow 130

Zeshan Xu, Jie Zhu, Li Yang and Chun Zuo
Mining the Relationship between Object-Relational Mapping Performance Anti-patterns and Code Clones 136

Haifeng Wang, Kun Yang, Xiangnan Zhao, Yuchen Cui and Weiwei Wang
Contribution-based Test Case Reduction Strategy for Mutation-based Fault Localization (S) 142

Anelis Pereira-Vale, Daniel Perovich and Sergio F Ochoa
Understanding the Pre-Contract Process of Small Software Projects 146

Junfeng Zhao, Wenhui Gai and Han Wu
Fortran Code Refactoring Based on MapReduce Programming Model 152

Jiman Du, Jiaxin Zhu, Hui Li, Wei Chen, Lijie Xu, Jie Liu and Zhifeng Chen
A Case Study of Dependency Network for Building Packages: The Fedora Linux Distribution 158

Shuo Li, Jie Liu, Haoxiang Tian, Shuai Wang, Wei Chen, Liangyi Kang and Dan Ye
EasyPip: Detect and Fix Dependency Problems in Python Dependency Declaration Files

Peiyang Li, Zixin Liu, Yuening Su, Hao Wang and Bo Jiang
CAPS: An Efficient Whole-Program Critical Paths Search Framework for Large-Scale Software

Zhiyong Ren and Hui Xu
Detect Stack Overflow Bugs in Rust via Improved Fuzzing Technique

Hao Zhu, Yichen Li, Jie Li and Xiaofang Zhang
An Efficient Design Smell Detection Approach with Inter-class Relation (S)

Tianmai Deng, Yixiao Xu, Chi Zhang and Zhengwei Qi
Dissecting Scale-Out Applications Performance on Diverse TLB Designs (S)

Yu Guo, Jun Guo, Mohan Shi and Aixuan Dong
Real-time Estimation Approach to Scrum Projects Effort: Task Point and Individual Reliability (S)

Zhiguo Li and Qin Li
ASC4AI: A Novel Automatic Service Composition Framework for AI Systems (S)

Omkarendra Tiwari and Vrinda Yadav
A Review of Methods for Identifying Extract Method Refactoring

Lufeng Wei, Zhi Li and Yuansong Qin
Towards generating contextualised fault scenarios for reliability risk analysis of cyber-physical systems (P)

Session SEAI: Software Engineering and AI

Zhenhua Zhang, Shizhan Chen, Guodong Fan, Guang Yang and Zhiyong Feng
CCGRA: Smart Contract Code Comment Generation with Retrieval-enhanced Approach

Ed van Doorn and Sylvia Stuurman
Detecting Design Patterns As Described By An Explicit Specification (S)

Jingjing Wang, Minhsuan Huang, Yuanping Nie, Xiaohui Kuang, Xiang Li and Wenjing Zhong
Fine-Grained Source Code Vulnerability Detection via Graph Neural Networks (S) 222

Xinyan Yi, Hao Xu, Lu Lu, Quanyi Zou and Zhanyu Yang

Software Defect Prediction via Positional Hierarchical Attention Network (S) 228

Ran Mo, Yushuo Wang, Yao Zhang and Zengyang Li

Just-in-Time Defect Severity Prediction (S) 232

Jiaxin Liu, Hao Xu, Lu Lu, Quanyi Zou and Zhanyu Yang

A Software Defect Prediction Method based on Multi-type Features and Feature Selection 238

Kirthy Kolluri, Tom Hill and Lawrence Chung

Identifying Risks and Risk Mitigation Strategies for Collaborative Systems during Requirements Engineering: A Goal-Oriented Approach (S) 244

Haochen Jin, Liwei Zheng and Zhanqi Cui

AFL2oop: Loop Coverage Guided Greybox Fuzz Testing 250

Liang Sun, Peng Wu and Shaoxian Shu

Research on Directed Grey-box Fuzzing Technology Based on Target (S) 256

Wei Li, Zhiyi Zhang, Yifan Jian, Chen Liu and Zhiqiu Huang

DeepRank: Test Case Prioritization for Deep Neural Networks 262

Shaojian Qiu, Shaosheng Wang, Yujun Liang, Wenchao Jiang and Fanlong Zhang

Code Clone Detection via Software Visualization Representation Learning 268

Guang Yang, Tianchong Jin and Liang Dou

Heterogeneous Directed Hypergraph Neural Network over abstract syntax tree (AST) for Code Classification 274

Shuaishuai Feng, Xi Wu, Yongxin Zhao and Yongjian Li

Dispatching and Scheduling Dependent Tasks Based on Multi-agent Deep Reinforcement Learning 280

Qing Qi and Jian Cao

GAT-Team: A Team Recommendation Model for Open-Source Software Projects 286

Yiheng Shen, Xiaolin Ju, Xiang Chen and Guang Yang

An Empirical Study of Adversarial Training in Code Comment Generation 292

Jun Cheng, Honglei Su and Jari Korhonen
No-Reference Point Cloud Quality Assessment via Weighted Patch Quality Prediction 298

Jie Li, Lixuan Li, Hao Zhu and Xiaofang Zhang

GraphPLBART: Code Summarization Based on Graph Embedding and Pre-Trained Model 304

Session KE: Knowledge Engineering

Qia Wang, Yi Wang and Ying Wang

An Efficient Lossless Graph Summarization Method for Large Streaming Graphs 311

Wen Sun, Yifan Li, Junfeng Yao, Qingqiang Wu and Kunhong Liu

Combining Structure Embedding and Text Semantics for Efficient Knowledge Graph Completion 317

Li-E Wang, Huachang Zeng, Shenghan Li, Xianxian Li, Shengda Zhuo, Jiahua Xie, Bin Qu, Zhigang Sun and Tianran Liu

CRCC: Collaborative Relation Context Consistency on the Knowledge Graph for Recommender Systems (S) 323

Yuxin Zhao, Xinjun Mao, Bo Sun, Tanghaoran Zhang and Shuo Yang

An Effective Method for Constructing Knowledge Graph to Search Reusable ROS Nodes (S) 329

Bin Li, Jian Wen Ding, Zhen Nan Chen, Han Peng Jiang and Fan Lin

Multi-Domain Feature Representation and Multi-Dimensional Feature Interaction for Person-Job Fit 333

Shizhao Huang, Yimin Wu, Jinwei Lu and Chao Deng

Related Questions Detection Model in Stack Overflow based on Semantic Matching 339

Yifan Wang, Jibin Wang, Jing Shang, Zhuo Chen, Xuelian Ding and Heyuan Wang

Learning Event Logic Graph Knowledge for Credit Risk Forecast 345

Junmin Zhu, Yizhao Yao, Xianwen Deng, Yaoguang Yong, Yanhao Wang, Libo Chen, Zhi Xue and Ruijie Zhao

SAWD: Structural-Aware Webshell Detection System with Control Flow Graph 351

Tong Niu, Yicong Xu, Shengping Xiao, Lili Xiao, Yanhong Huang and Jianwen Li

LTLf Satisfiability Checking via Formula Progression (S) 357

Yifan Wang, Masaki Nakamura, Kazutoshi Sakakibara and Yuki Okura
Formal Specification and Verification of an Autonomous Vehicle Control System by the OTS/CafeOBJ method (S)  
Magali Legast and Axel Legay
Assistance in the management of rule sets for rule-based expert systems (S)  
Mengyuan Wang, Tengfei Li, Jing Liu, Hui Dou, Hongtao Chen, John Zhang and Lipeng Zhang
Modeling and Verification of Autonomous Driving Systems under Stochastic Spatio-Temporal Constraints

Session DPCC: Data Privacy and Cloud Computing

Georg Grabner, Ali Ahmed and Nilufar Baghaei
Using Blockchain to Preserve Chain of Custody: Cloud Forensics Analysis (S)  
Wenxiu Xu, Ningjiang Chen and Huan Tu
DNN Inference Task Offloading Based on Distributed Soft Actor-Critic in Mobile Edge Computing

Yuxin Xiang, Bing Deng, Hongyu Zhang, Randy Xu, Marc Mao, Yun Wang and Zhengwei Qi
Bindox: An Efficient and Secure Cross-System IPC Mechanism for Multi-Platform Containers

Xiguo Gu, Huiwen Yang, Shifan Liu and Zhanqi Cui
Smart Contract Vulnerability Detection Based on Clustering Opcode Instructions

Yingjia Du, Yuan Fei, Sini Chen and Huibiao Zhu
Formalization and Verification of Data Auction Mechanism Based on Smart Contract Using CSP

Session DMA: Data Mining and Analytics

Gabriel Mac'Hamilton Renaux Alves and Alexandre Magno Andrade Maciel
Survey on Data Ingestion for AutoML (S)

Bowen Chen
DGNN: Dynamic Graph Neural Networks for Anomaly Detection in Multivariate Time
Dolly Agarwal, Abdullah Al Imran and Kasun S Perera
FOODIE: A Data-centric Sifting Framework for Social Media Analytics

Jie Xiong, Huiyuan Li, Qiyuan Duan and Qihan Du
Towards Extreme Multi-label Text Classification Through Group-wise Label Ranking

Guangwei Zhang, Ming Jing, Yunjing Liu, Li Zhang, Anming Dong and Jiguo Yu
Interactive Visualization of Temporal Brain Connectivity Data based-on Frequent Feature Mining (S)

Peng Zhang, Zhaohui Zhang, Chaochao Hu and Pengwei Wang
A Dynamic Drilling Sampling Method and Evaluation Model for Large-Scale Streaming Data

Bowen Chen
ASTD: Automatic Seasonal-Trend Decomposition for Time Series

Lidan Zhao, Tong Li, Zhen Yang and Junrui Liu
A novel subjective bias detection method based on multi-information fusion

Rongxi Guo, Wanrong Jiang, Xi Zhao, Ding Cao and Guiquan Liu
ASMix: An Attention-based Smooth Data Augmentation Approach

Samuel Abedu, Solomon Mensah, Frederick Boafo, Eva Bushel and Elizabeth Akuafum
An Optimal Spacing Approach for Sampling Small-sized Datasets

Wei Du and Shifei Ding
Heterogeneous Multi-Agent Communication Learning via Graph Information Maximization

Ziyi Zhang, Rong Yan and Wei Yuan
Identifying Influential Spreaders in Complex Networks Using Neighborhood Network Structure

Session OAEP: Optimization Approaches for Enhanced Performances

Bishshoy Das, Milton Mondal, Brejesh Lall, Shiv Dutt Joshi and Sumantra Dutta Roy
Altering Backward Pass Gradients to Improve Convergence (S) 481

Wei Liang, Wanghua Shi, Zhengyu Liao, Shiyou Qian, Zhonglong Zheng, Jian Cao and Guangtao Xue

BOP: A Bitset-based Optimization Paradigm for Content-based Event Matching Algorithms (S) 487

Yu Zhang, Ningjiang Chen and Siyu Yu

Transient Data Caching Based on Maximum Entropy Actor-Critic in Internet-of-Things Networks 493

Yulong Wang, Kai Deng, Fanzhi Meng, Zhi Chen, Mingyong Yin and Run Yang


Yaodanjun Ren, Chen Chen and Zhengwei Qi

HaeNAS: Hardware-Aware Efficient Neural Architecture Search via Zero-Cost Proxy 505

Rong Wu

L*-Based Learning of Probabilistic Timed Automata with One Clock (S) 511

Session MLDLA: Machine Learning and Deep Learning Applications

Shunwen Shen, Lvqing Yang, Sien Chen, Wensheng Dong, Bo Yu and Qingkai Wang

DeepMultiple: A Deep Learning Model for RFID-based Multi-object Activity Recognition 518

Shuai Li, Bing Shi and Yaping Deng

A Dynamic Matching Time Strategy Based on Multi-Agent Reinforcement Learning in Ride-Hailing 524

Zhennan Chen, Zhicheng Zhang, Pengfei Li, Lingyue Wei, Shibo Feng and Fan Lin

mTrader: A Multi-Scale Signal Optimization Deep Reinforcement Learning Framework for Financial Trading (S) 530

Lei Mao, Jianxia Chen, Shi Dong, Liang Xiao, Haoying Si and Xinyun Wu

Aspect-level Sentiment Analysis Based on Convolutional Network with Dependency Tree 536

Georg Stettinger and Franz Wotawa
Smart Monitoring for Safety-Assurance in Autonomous Driving (S) 542

*Xingchen Ding, Chong Teng and Donghong Ji*

Fake News Detection with Context Awareness of the Publisher 548

*Bakhta Haouari, Rania Mzid and Olfa Mosbahi*

PSRL: A New Method for Real-Time Task Placement and Scheduling Using Reinforcement Learning 554

*Rui Xu, Cheng Zeng, Yujin Liu, Peng He and Min Chen*

A Triplet Network Approach for Chinese Confusing Text Classification 560

*Xingfu Li, Bangchao Wang, Hongyan Wan, Yang Deng and Zihan Wang*

Applications of Machine Learning in Requirements Traceability: A Systematic Mapping Study (S) 566

*Chen Qian, Hui Xiong and Jiyun Li*

FeDeFo: A Personalized Federated Deep Forest Framework for Alzheimer's Disease Diagnosis 572

*Kang Zhong, Wu Guo, Bin Gu, Weitai Zhang and Chao Lin*

A Detection-based Attention Alignment Method for Document-level Neural Machine Translation 578

*Gaoyi Lin, Zhihua Zhang and Zhanqi Cui*

Widget Hierarchy Graph Guided Crash Reproduction Method for Android Applications (S) 584

*Xing Zhang, Tinglong Tang, Yirong Wu and Tingwei Quan*

Construction Site Fence Recognition Method Based on Multi-Scale Attention Fusion ENet Segmentation Network (S) 588

*Jingxia Zeng, Mianfan Chen, Jianing Liu and Yuncheng Jiang*

SLPKT: A Novel Simulated Learning Process Model for Knowledge Tracing 594

*Chen Qian, Boyin Yang and Jiyun Li*

SARNet: A Self-Attention Embedded Residual Network for Multiclass Classification of Chronic Wounds (S) 600

*Yimin Peng, Rong Hu and Yiping Wen*

AT-NCF: An Attention-based Time-aware Neural Collaborative Filtering Approach for Personalized Recommendation (S) 604
Session LMQA: Language Models and Question Answering

Aoxing Wang, Pengfei Duan, Yongbing Li, Wenyan Hu and Shengwu Xiong
LMGFuse: Language Models and Graph reasoning Fuse deeply for question answering

Zuotong Xie, Kai Ouyang, Xiangjin Xie, Hai-Tao Zheng, Wenqiang Liu, Dongxiao Huang and Bei Wu
KLAPrompt: Infusing Semantic Knowledge into Pre-trained Language Models by Long-answer Prompt Learning

Xingang Wang, Xiaomin Li and Xiaoyu Liu
Reinforced Multi-modal Circulant Fusion based Transformers for Rumor Detection (S)

Qian Wang, Cheng Zeng, Yujin Liu and Peng He
A Medical Question Classification Approach Based on Prompt Tuning and Contrastive Learning (S)

Xiaoyan Gao, Ji Chen, Xiangqian Ding, Shuo Shi, Junhua Jia and Xiaowei Xin
An intelligent question answering system based on RoBERTa-WWM under home appliance knowledge graph (S)

Yingshuai Kou, Neng Gao and Yifei Zhang
Variational Autoencoder with Stochastic Masks to Solve Exposure Bias in Recommendation (S)

Fengyong Peng, Xi Wu, Yongxin Zhao and Yongjian Li
Anaphora Ambiguity Detection Method Based on Cross-domain Pronoun Substitution (S)

Zhe Wen, Yatao Qi, Huan Liu, Zheng Lin and Weiping Wang
Multi-source Machine Reading Comprehension with Meta-Learning and Adaptive Adversarial Training

Tong Zhang, Ailing Tang and Rong Yan
Enhanced BERT with Graph and Topic Information for Short Text Classification (S)

Xi Tao, Hao Wang, Xiangfeng Luo and Pinpin Zhu
Few-Shot Link Prediction using Variational Heterogeneous Attention Networks 660

Xingchen Ding, Chong Teng, Donghong Ji and Fei Li
HGAPT: Heterogeneous Graph Augmented Prompt Tuning for Low-Resource Fake News Detection 666

Yunlai Luo
Token-level Input Grammar Synthesis (P) 672

Session CVA: Computer Vision and Applications

Puhui Qu, Keyu Chen and En Cheng
YOLOv7-marine: An Improved YOLOv7 Model for Object Detection in Marine Environments (S) 675

Yuheng He, Wenbin Yang, Xiangfeng Luo, Liyan Ma and Shaorong Xie
Object-Level Contrast Learning for 3D Sparse Object Detection in Ocean Scene 679

Qiaoning Lei, Yinsai Guo, Liyan Ma and Xiangfeng Luo
Few-Shot Object Detection via Instance-wise and Prototypical Contrastive Learning 685

Jiyun Li, Sheng Yang and Qian Chen
SWS-NET: An Image Segmentation Framework For Chronic Wounds Based On Self-Supervised Learning 691

Shouhong Wan, Rui Wang, Peiquan Jin and Xuebin Yang
BDC-FR: Faster R-CNN with Balanced Domain Classifier for Cross-Domain Object Detection (S) 697

Luo Yang and Lin Rongheng
MBR-MDA: Multi-person Behavior Recognition Method Based on Multi Descriptors Aggregations 701

Notes: (S) denotes a short paper. (P) denotes a poster, which is not considered a paper.
Special Session SCE: Social Computing and Engineering
Abstract—In the current social environment, social crisis events occur frequently with significant impacts. Group intention mining through automatic technologies for managing social crises has gained extensive attention. This paper presents an overview of research on group intention mining in social crisis events, covering three areas: knowledge graph inference, intention attribution, and risk management. Knowledge graph inference facilitates the detection of group intention in crisis events. It is supported by the construction of crisis knowledge graphs, which organize crisis elements and inter-element relations into structured semantic knowledge. The interpretable semantics in the crisis knowledge graphs enables attribution of intention. Group intention mining consists of intention detection and intention attribution, serving the risk management of social crisis events. To gain insights into the process of group intention mining in social crises, the Covid-19 event is selected as a case study. Finally, the paper proposes future research directions to solve the limitations of existing intention mining methods in social crises.

Index Terms—Social Crisis Events, Intention Mining, Intention Explanation, Risk Management

I. INTRODUCTION

Social crises are complicated and dynamic events that threaten public safety and interest, influenced by various interconnected factors. The Covid-19 pandemic serves as a prime example of a global health crisis that has significantly disrupted the everyday lives of individuals. Exploring the formation and evolution of social crises is crucial to safeguard public safety. However, in the complex international environment, social crisis management is challenging for all countries. The process of managing social crises consists of three stages: pre-crisis detection, real-time crisis response, and post-crisis review, as illustrated in Fig. 1. To effectively manage a crisis, it is crucial to mine the intention of groups involved, as it provides valuable insights into their behavior in a given situation. For example, tracking the public’s sentimental intention during the Covid-19 pandemic can help restore social development. The model of group intention mining in social crisis events is defined as $I_{\text{crisis}} = \{I_{\text{det}}, I_{\text{att}}\}$, where $I_{\text{det}}$ is intention detection, $I_{\text{att}}$ indicates intention attribution. However, intention mining during social crises is challenging due to the unclear evolutionary trajectory and elusive precursors of such events. On the one hand, the vast amount of knowledge in social crises is both extensive and sparse, resulting in a semantic imbalance arising from long-tailed event information. On the other hand, the semantics of words may evolve over time, and new words or phrases keep emerging, making it difficult to analyze crisis-related data. Additionally, using unexplainable intention decision models in analyzing social crisis events involving life safety poses a risk.

This paper aims to address the issues related to group intention mining in social crises. It provides a comprehensive overview of current research in three closely related areas: knowledge graph inference, intention attribution, and risk management for social crisis events. Section II covers knowledge graph inference, which supports the identification of group intention in crisis events. Section III discusses intention attribution, which helps to determine the motivations and goals of crisis actors. Group intention mining, encompassing intention detection and attribution, plays a critical role in social crisis risk management, as discussed in Section IV. Section V presents a case study on Covid-19, demonstrating how the integration of knowledge inference and intention attribution can support risk management during a social crisis. Finally, the paper proposes improvements to overcome the limitations of existing research on intention mining in social crises.

II. KNOWLEDGE GRAPH CONSTRUCTION AND INFERENCE FOR SOCIAL CRISIS EVENTS

The extensive and sparse nature of social crisis data poses a challenge for effective knowledge representation. Knowledge graph offers a viable solution for organizing and integrating discrete semantic knowledge related to social crises. Using the constructed crisis knowledge graph for knowledge inference enables the detection of group intention.

Knowledge graph improves the agencies’ ability to deal with crises in different areas. In counter-terrorism detection, to identify terrorist organizations, Bangerter et al. [1] constructed a knowledge graph from the Global Terrorism Database and trained graph neural network (GNN) by inductive link prediction technique. Yang et al. [2] utilized the co-occurrence matrix and relationship knowledge graph to sort out the development tendency of various topics in counter-terrorism intelligence. In public health domain, to identify and explain misleading Covid-19 statements on social media, Kou et al. [3] integrated the knowledge of both experts and non-experts to construct a Covid-19 knowledge graph. In environmental protection, Liu et al. [4] built a multi-source oil spill detection knowledge graph using rule reasoning and GNN...
When utilizing knowledge graphs for crisis management, three steps must be taken, including knowledge extraction, fusion, and inference. To overcome challenges encountered during these stages, researchers have proposed several solutions. In the knowledge extraction stage, to solve the entity overlap problem during the triplet extraction, Wei et al. [7] regarded the triplet extraction as a discrete classification problem, extracting entities first and then identifying entity relation. Aiming at the long-term dependence problem of triples, Ye et al. [8] proposed a generative model to generate entity-relation-entity successively. Targeting the problem of missing part of the contextual information for extracted triples, Geng et al. [9] presented a convolutional recurrent neural network based on attention mechanism to jointly uncover entities and relations. In the knowledge fusion stage, Trisedya et al. [10] introduced an entity alignment model based on attribute-embedded characters and transitivity rule. Wu et al. [11] proposed a cross-lingual entity matching model named CLEM, which integrates multimodal information embedding matching entities based on a multi-perspective spatial learning method. In the stage of knowledge inference, given the incompleteness of knowledge graph, Lei et al. [12] proposed knowledge graph completion and reasoning based on symbolic methods and reinforcement learning. Niu et al. [13] presented a coding and decoding model based on GNN and high-dimensional structure weight. Knowledge graph has been applied in domain knowledge transfer, target recognition, and semantic tracing. The improvement of knowledge graph construction approaches promotes knowledge graph’s application in social crisis management. Based on relevant research, the framework of knowledge graph construction and inference in social crises is shown in Fig. 2(a). Firstly, through the knowledge extraction, the triad of crisis elements is obtained as \( KT = \{ e_h, r, e_l \} \), where \( e_h \) is a header event element, \( e_l \) is a tail event element, and \( r \) denotes the relation between the above two elements. Secondly, the crisis knowledge graph \( CG \) is gained through the knowledge fusion, which is expressed as \( CG = \{ E, R \} \), where \( E \) is the set of crisis elements, \( R \) is the set of \( r \). Finally, based on the crisis graph and the target event data, the group intention is derived by knowledge inference as \( I_{det} = f(CG, D) \), where \( D \) is the target event data, \( f \) denotes the knowledge inference function. Knowledge graph as a way of structured representation have the ability to manage complex crisis data.

Current research on knowledge graph has made notable advancements in detecting intentions related to specific domains of risk. However, challenges arise when attempting to apply these approaches to cross-cutting social crises. These difficulties stem from the complexity and semantic co-reference of elements within crisis events. As a result, the constructed social crisis knowledge graph is often semantically sparse, failing to accurately capture the continuous scene semantics involved in social crisis events.

III. INTENTION ATTRIBUTION AND EXPLANATION FOR SOCIAL CRISIS EVENTS

In recent years, deep learning has been applied in various public places to ensure public safety. However, the poor interpretability of deep models impairs their reliability and trustworthiness, especially in social crises involving the safety of people’s lives and property. Therefore, it becomes imperative to explain and attribute public intention of social crises in a human-understandable way.

In different application domains, researchers have explored the interpretability of deep models. In the medical field, Assegie et al. [14] adopted LIME and SHAP to rank the importance of features and explain the model’s output on whether a patient is diabetic or not. To provide clinical doctors with a clear understanding of the classification criteria utilized by GNN in Alzheimer’s disease prediction, Anjomshoae et al. [15] proposed a single node classification explanation method. By scrutinizing the alterations in output arising from decomposed input values, the extent of input values’ impact on predictions can be gauged. In the finance domain, Pisoni et al. [16] discussed how insurance companies provide cus-
customers with suggestions and explanations on the recommended amount of insurance coverage. In the industrial area, Zhang et al. [17] utilized SHAP to identify salient features in predicting diagnostic faults in power transformers. In the environment area, Cilli et al. [18] employed random forest model with Shapley value to detect the driving factors that contribute to the occurrence of fires. In the sports field, to analyze the trend of NBA games, Wang et al. [19] applied random forest and feedforward neural network to build a prediction model. Then, they adopted the LIME model to explain the prediction.

To increase trust in deep models across diverse domains, it is crucial to continuously enhance intention attribution techniques. Automated approaches like dependency graphs, feature engineering, and alternative models have been developed for intention explanation. Semantic barriers exist between different modal data, targeted attribution algorithms have been proposed to verify diverse data attributes. For image data, various visual interpretation methods have been introduced, including LIME, GRAD-CAM, and RISE. The SIDU method [20], addressed salient region localization by creating pixel similarity difference and uniqueness masks extracted from the last convolutional layer of the convolutional neural network. For textual data, to predict and explain the sentiment in speech, Zhang et al. [22] proposed the RexNet model and the XAI perceptual processing framework inspired by the perceptual process of cognitive psychology, in which contrast salience, counterfactual synthesis, and contrast cue interpretation are treated as interpretation methods. Of course, there are also many interpretable tactics for multimodal data. To help non-expert end users understand the decision-making process of intelligent agents, Muñoz et al. [23] utilized a success probability-based approach to construct a humanoid explanation that visually displays the state of the autonomous robot after taking an action. To explain the driving route of autonomous vehicles in complex environments, Zhang et al. [24] propose a Multimodal Trajectory Prediction Transformer model to retrieve the influencing factors of the prediction. Based on related research, the research framework for group intention explanation is shown in Fig. 2(b). For example, based on the interpretable crisis community knowledge graph, the interpretation of group intention $I_{det}$ is gained as $I_{att} = g(CG, I_{det})$, where $g$ is the knowledge graph-based attribution method. The enhancement of attribution techniques has facilitated their utilization in various vertical domains.

Existing intention attribution methods are effective in providing understandable explanations. However, the presence of fuzzy crisis elements and the complex mechanisms of action among these elements present a notable obstacle in identifying the factors that determine group intention and behavior.
IV. RISK MANAGEMENT FOR SOCIAL CRISIS EVENTS

Social crisis data involves numerous risk elements, with intricate interdependencies among them. Intelligent technologies equipped with robust data analysis capabilities exhibit superior efficacy in identifying and mitigating social crises compared to human operators. These technologies leverage advanced techniques such as machine learning, data mining, and natural language processing to analyze vast amounts of data, identifying patterns and insights that might otherwise be missed by humans.

Organizations responsible for crisis management rely on a combination of management theory and intelligent technology to detect and respond to social crises. Mcgowran et al. [25] utilized a disaster risk detection method and portfolio theory to analyze the overall response approach. Wei et al. [26] developed an intelligence model for social crisis early warning based on the intelligence production chain. Liu et al. [27] fused information collection, epidemic monitoring, and risk assessment theories of epidemic risk to enhance the public health emergency response system’s capacity. Yan [28] proposed the establishment of an efficient mechanism for sharing crisis response information among international actors to address abrupt environmental crises across various regions. Simpson et al. [29] identified forms of interactions that generate risks and subsequently integrated corresponding response strategies into a climate change risk framework to enhance decision-making.

In response to the increasing demand for risk management, there has been a growing interest in the development of automatic technologies. Researchers have proposed numerous data analysis methods to tackle crisis events. Zhu et al. [30] utilized K-means to construct an anti-crime information system to predict potential crime hazards. Deng et al. [31] developed a spatiotemporal hotspot-factor model to study the temporal and spatial locations of unusual crime events. Guo et al. [32] used video reconstruction to locate security threats. Zhong et al. [33] developed a security risk assessment system for sporting events using neural networks. Li et al. [34] studied public risk perception and emotion expression during the Covid-19 pandemic to assist in managing public health risks. To normatively organize various multi-source heterogeneous mass event information, Ren et al. [35] extracted event elements from mass event data based on the BiLSTM-CRF model, constructed a mass event knowledge graph reflecting the correlations among the event elements. Based on the comprehensive related research, the framework of social crisis risk management is shown in Fig. 2(c). In this framework, data analysis methods such as knowledge graph are applied to encode crisis data and decode group intention. At the same time, interpretable semantic information in the knowledge graph provides support for risk intention attribution.

Previous research has shown success in solving crisis events within specific fields through various methods. However, social crisis events are complex, encompassing diverse domains. Current methods lack the ability to comprehensively incorporate complex event element clues and multi-source semantic knowledge, leading to a one-sided and incomplete treatment of social crisis-related information processing.

V. CASE STUDY AND ANALYSIS: THE COVID-19 EVENT

The Covid-19 event is treated as a case to mine sentimental intention of group in social health crisis events. Specifically, the public opinion knowledge graphs are constructed to organize the sentiment information in the crisis event, and the potential group sentimental intention is detected through knowledge inference. The identified sentiment is further explained through the distribution of feature words within corresponding knowledge graphs of different communities. The implementation process of the Covid-19 event consists of three parts: data source, public opinion knowledge graph construction, and group intention analysis.

a) Data source: The data of the Covid-19 incident was obtained by crawling official news media reports from January to July 2020 and from April to August 2021. After preprocessing, a total of 26,192 news samples were collected.

b) Public opinion knowledge graph construction of the Covid-19 event: Firstly, the adjectival words in the corpus reflecting sentiment are extracted using Jieba¹. Secondly, Word2Vec² is used to train the public opinion word vectors. For a public opinion word \( w \), its word vector \( t_{pow} \in \mathbb{R}^{dim} \) is

\[
t_{pow} = W_{em}w_{pow}
\]

where \( dim \) is vector dimension, \( W_{em} \) is embedding matrix, and \( w_{pow} \) denotes the one-hot vector of \( x \). Cosine similarity method is employed to compute inter-word similarity. For two public opinion word vectors \( t_{pow1} \) and \( t_{pow2} \), their similarity is shown below,

\[
sim(t_{pow1}, t_{pow2}) = \frac{\sum_{i=1}^{dim} t_{pow1i} \cdot t_{pow2i}}{\sqrt{\sum_{i=1}^{dim} t_{pow1i}^2} \cdot \sqrt{\sum_{i=1}^{dim} t_{pow2i}^2}}
\]

Finally, based on the word vectors and inter-word similarity, the public opinion knowledge graphs are obtained using Gephi³. Fig. 3 displays several month-level graphs, each with representative opinion words shown underneath. These opinion words serve as explanation \( I_{att} \) for group intention.

c) Group intention analysis of the Covid-19 event: Firstly, Sentiment score of public opinion words being positive or negative is calculated utilizing a Bayesian model:

\[
P(pos|x) = \frac{P(x|pos) \cdot P(pos)}{P(x|pos) \cdot P(pos) + P(x|neg) \cdot P(neg)}
\]

when \( P(pos|x) > 0.5 \), the word is positive, otherwise it is negative. Then, the positive and negative sentiment intensity for that month was gained based on the frequency of positive or negative words in the month-level corpus data:

\[
s_{pos} = \frac{count(pos)}{count(pos) + count(neg)}
\]

¹https://pypi.org/project/jieba
²https://pypi-python.org/pypi/word2vec
³https://gephi.org
where $\text{count}(\text{pos})$, $\text{count}(\text{neg})$ are the word frequencies of positive and negative words respectively, $s_{\text{pos}}$, $s_{\text{neg}}$ are the sentiment intensity of positive and negative feelings respectively. Finally, the sentiment intensity of the Covid-19 event is represented in Fig. 6. It can be observed that group intention $I_{det}$ is classified into three categories: sentiment depression, sentiment optimism, and sentiment fallback intention. The corresponding opinion knowledge graphs for each of these categories are presented in Fig. 3, Fig. 4, Fig. 5, respectively.

Sentiment depression intention(2020.1-2020.7): Covid-19 outbreak caused an increase in pneumonia cases and global fear. Rapid response of the government controlled overall public opinion. Sentiment optimism intention(2020.7-2021.6): effective epidemic prevention measures and good deeds led to peak in positive sentiment intensity. Sentiment fallback intention(2021.6-2021.8): negative sentiment increased due to mutant strain and economic downturn, but still remained below the level of positive sentiment.

In general, through collaborative efforts between governmental authorities and media outlets, the public’s outlook on the coronavirus has progressively improved, resulting in a proactive resumption of their daily routines.

VI. CONCLUSION AND FUTURE RESEARCH

Group intention mining is a crucial aspect of managing social crises effectively. This paper presents a comprehensive overview of current research on group intention mining, including knowledge graph inference, intention attribution, and risk management. However, current methods still exhibit limitations that require further investigation in the future.

To address the challenge of managing the all-round and multifaceted risk information in social crisis events, crisis knowledge graphs can be constructed based on distributed representation. This involves treating noun semantic clusters as entities and verb semantic clusters as relations, resulting in a crisis knowledge graph that utilizes adaptive synonymous semantic expressions.

To tackle the challenge of identifying comprehensive group intention in social crises with diverse and obscure elements, event information and crisis knowledge graphs are fused to mine potential group intention. Different encoders encode event descriptions, background knowledge, and common sense, followed by an attention-based intention decoder for intention extraction.

To solve the difficulty of tracing the evidence for determining group intention due to the complex action mechanism...
between crisis elements, the attribution and reflection of group intentions are realized based on explainable methods such as knowledge graph. The reverse optimization process of model is designed to maximize the current intention, and reflection method is utilized for intention re-determination.

Social crises endanger the overall well-being and common interests of the entire society. This paper utilizes interpretable knowledge graph and intention mining techniques to analyze the process of group intention mining in social crisis management. It is hoped that our research provides constructive insights for social crisis management agencies.

References


Topic and Speaker-aware Hierarchical Encoder-Decoder Model for Dialogue Generation

Qitao Hu, Xiao Wei*

School of Computer Engineering and Science, Shanghai University, China
xwei@shu.edu.cn

Abstract—As one of the most common social behavior in human society, communication in multi-turn conversation or dialogue system has always been a research focuses of natural language processing (NLP). The quality of downstream tasks in multi-turn dialogue is often determined by the result of dialogue context modeling. For dialogue generation, the context information will determine the consistency and diversity of the generated responses. However, the current research on dialogue generation increasingly relies on external information rather than mining from the dialogue content itself. In this paper, we propose a topic and speaker-aware hierarchical encoder-decoder (TSHED) model to capture the topic and speaker information flow in the context for response generation with the hierarchical transformer-based framework. Specifically, we obtain semantic information of each utterance at word-level and then apply topic and speaker-aware attention to model context at utterance-level. Experimental results on two open-domain datasets show that TSHED significantly improves the quality of responses and outperforms strong baselines.

Index Terms—open-domain dialogue system, dialogue generation, encoder-decoder model, transformer

I. INTRODUCTION

Daily conversation is one of the crucial components of human social activities. Most daily conversations among groups of people can be summarized as open-domain dialogue or chit-chat, which are multi-turn and informative. From the perspective of dialogue content, the context is approximately equal to historical utterances in previous turns. Therefore, some studies focus on multi-turn conversations with context modeling for the downstream task of the dialogue system, such as dialogue generation, dialog emotion detection [1], abstractive dialogue summarization, etc. For dialogue generation, it is vital to make full use of context to generate consistent and logical responses. The earlier deep learning-based methods focus on short-text conversation or local context [2], [3]. Nevertheless, context modeling is not only a process of word alignment or sentence alignment, but also a process of perceiving the information flow that drives the dialogue. Serban et al. [4] proposed a ground-breaking hierarchical encoder-decoder framework called HRED to model context. HRED consists of a word-level encoder that encodes utterances, an utterance-level encoder that maps each utterance representation into dialogue context and a decoder that generates tokens. Since then, the ideas about context modeling with hierarchical architecture have been widely used in multi-turn dialogue generation tasks [5]–[7].

*Corresponding author: Xiao Wei, xwei@shu.edu.cn
DOI reference number: 10.18293/SEKE2023-046

However, compared with documents, the topic flow in dialogue is more complex, which means the topic density of dialogue is higher and the topic shift in dialogue is more unpredictable. In addition, the speaker flow is unique to dialogue compared with other natural language scenes. The participants in the dialogue are at least more than one, which makes different interlocutors in the same scene with different response ideas. In our work, we introduce two granularity of topics, called dynamic topic which means the topic that arises as the conversation progresses and general topic which means the topic that reflects the phases of the dialogue process. As shown in Fig.1, this is the daily conversation between two middle-aged women who meet and chat. From the perspective of the topic, their topics after greeting are pretty random, such as ‘appearance’ and ‘family’, but their dialogue content follows the fixed pattern of greeting, chatting and saying goodbye. From the perspective of the speaker, ‘B’ shows a polite and warm attitude. The above instance shows that the topic and the
speaker are essential supplements to the context information in a multi-turn conversation scenario, affecting the quality of the generated response. In recent research on topic-aware response generation, the use of the topic includes extracting topic words or using external knowledge. Zhang et al. [8] took meaningful words that appeared in the dialogue history and response as topic words to generate topic-relevant responses. Feng et al. [9] used topic-aware attention and external knowledge sources to promote the understanding of dialogue history. In terms of the speaker, recent work uses external information about the speaker to model. Majumder et al. [10] used commonsense knowledge bases to expand and enrich persona descriptions.

However, there is still internal mining space in dialogue history rather than just external information to supplement the context in dialogue context modeling. In this paper, we propose a new model called TSHED to fully use the dialogue history with a multi-degree attention mechanism in context modeling for response generation. The core idea is applying several context-aware attention to model context at the utterance level. Specifically, we improve the traditional text segmentation algorithm TextTiling [11], complete the segmentation of two granularity of topics, and then use the transformer-based word-level encoder to get the representation of each utterance. After that, the utterance-level encoder obtains the representation of the whole context via topic-aware attention and speaker-aware attention. Finally, the GRU decoder is used to generate response token by token in combination with the attention weight. Experiments on two open-domain datasets demonstrate the effectiveness of TSHED, and further analyses reveal the advantage of TSHED in achieving state-of-the-art performances on the datasets.

II. RELATED WORK

As one of the hot topics in natural language processing (NLP), dialogue systems which are highly coincident with human interaction have attracted much attention in recent years. In particular, dialogue generation in dialogue system is a challenging task, the research focus of which has gradually changed from single-turn to multi-turn. One reason is that multi-turn conversations are more common in daily life. Longer dialogue turns mean more complex context composed of speaker messages which determines conversation topics, speaker goals and style [5]. How to model context to generate better responses is a challenge in this area.

Earlier Studies often used representations of words in dialogue history accumulated to contextualize or focus on the recent context [2], [3]. Serban et al. [4] proposed Hierarchical Recurrent Encoder-Decoder (HRED), which built a ground-breaking hierarchical encoder-decoder framework to achieve context-awareness in dialogue systems. Since then, hierarchical-based models have been widely studied in the task of multi-turn dialogue generation. However, the information distribution of the context is uneven, and it is obviously inappropriate to treat all contexts equally. Therefore some researchers try to introduce the attention mechanism. Tian et al. [12] compared the non-hierarchical model with the hierarchical model. They proposed WSeq finding that neural networks can produce longer, more meaningful and diverse responses with more context information. Zhang et al. [13] introduced a self-attention mechanism to capture long-distance dependencies. With the extensive use of pre-trained language models in natural language generation tasks [14], it is possible to obtain richer semantic information via various pretraining objectives. Gu et al. [15] employed a hierarchical Transformer architecture with two training objectives to capture hierarchical coherences on dialogue generation tasks. However, the role of topic in dialogue context modeling had been almost ignored in the above research. In addition to implicit topic modeling with latent variables [5], [7], Yoshikoshi et al. [16] used keywords or topic entities based on dialogue history to explicit topic modeling. Nevertheless, the topic dominated by utterances may not be represented by variables or a word. Inspired by the above research work, we transfer dialogue history into information blocks from the perspective of topic and speaker, and then combine topic-aware and speaker-aware attention mechanisms into hierarchical encoder-decoder framework to model meaningful and informative context for the task of response generation.

III. METHODOLOGY

Our model consists of two parts: (1) Multi-granularity topic segmentation, which is used to segment the multi-turn dialogue history with topic boundaries in an unsupervised way. (2) TSHED Network, which uses topic and speaker-aware attention with hierarchical architecture to achieve context encoding, and finally decodes to generate responses.

A. Topic Segmentation

Compared with documents [17], dialogues are less structured, more organized and promoted by topics. We call topic flow hidden under the dialogue. In our paper, topic flow is not unique because topics are multi-granularity. For example, in a daily conversation among colleagues, the common topic flow of “greeting → work → psychology → vacation” can be abstracted from it. The trend of topic flow after the topic “vacation” can lead to any others, so we call this kind of topic flow in a dialogue as dynamic topic flow. Besides, some topics that represent the rhythm of conversation are common to most dialogues, such as “introduction” or “end” which means the beginning or end of a dialogue; we call this kind of topic flow in a dialogue as general topic flow.

Such topic flow can be helpful in understanding the process of dialogue and the consistency of response generation. Given a continuous multi-turn dialogue $D = \{u_1, u_2, \ldots, u_n\}$, where $n$ is the number of utterances, we combine classic unsupervised topic segment algorithm, TextTiling [11] that focus on the change of topic similarity in the text, with sentence represented by pretrained Sentence-BERT [18], to divide multi-granularity topic boundaries. The differences from traditional TextTiling are shown in the Fig. 2(b), first, we keep the integrity of the sentence instead of dividing the text unit according to the number of words with fixed length and
take the utterance as the text unit; second, we dynamically maintain the unbalanced length of text blocks (formed by several text units) on both sides of token-sequence gap to obtain a more realistic token-sequence gap; third, we segment multi-granularity topics with the perspective of dynamic and general topics.

To be specific, each utterance $u_i$ is encoded as a sentence embedding $se_i$ by Sentence-BERT, then the similarity sequence is obtained by calculating the similarity combination of blocks with different lengths, which can be used to convert into two sequences; one is used to calculate the depth score sequence for dynamic topic sequence denoted as $Topic_d = [t_1^1, t_2^2, t_3^3, t_4^4, t_5^5, \ldots, t_n^n]$, the other is used to calculate the compression similarity sequence for local general topic sequence denoted as $Topic_g = [t_1^1, t_2^2, t_3^3, t_4^4, t_5^5, \ldots, t_n^n]$, where $t_s^i$ is the topic $s$ of $u_i$ and we call the gap between $t_i$ and $t_j$ as topic boundary, $y - 1$ is the max number of general topic boundaries. Then global general topic sequence denoted as $Topic_g$ can be obtained by calculating the index average of each topic boundary in $Topic_g$, which is as general topic sequence $Topic_g$ finally.

### B. TSHED Network

Compared with the document, there is another prominent feature in dialogue: more than one speaker participates in dialogue. Different speakers’ rhythms or narrative characteristics make the text features of conversation diverse. Thus, we design a Topic and Speaker-aware Hierarchical Encoder-Decoder model TSHED, which encodes utterances from word-level to utterance-level by a hierarchical architecture as shown in Fig. 2(a). During utterance-level encoding, TSHED considers the segmentation of different speakers into context while referring to the multi-granularity topic mentioned above. For any multi-turn dialogue, we denote the dialogue as $D = \{u_1, u_2, \ldots, u_n\}$, the dynamic topic segmentation of $D$ as $Topic_d$, the general topic segmentation of $D$ as $S_{sp} = Topic_g$, the speaker segmentation of $D$ as $S_{sp} = [s_1^n, s_2^1, s_3^3, \ldots, s_n^n]$.

1) **Word-level Encoder**: The word-level encoder we used in TSHED is transformer-based, designed to extract the semantic information of each utterance in the dialogue history $D$. For each word, we use the fixed dimension of the trainable embedding matrix $E_{w}$ to represent and the embedding $ew_{ij}$ corresponding to the $i$-th word in the $j$-th utterance. Then for each word embedding, we use the standard transformer encoder to encode, by taking the contextual embedding of the last token of each utterance (In our model, the special token $\langle eou \rangle$) to represent every utterance individually. We denote the output of the word-level Encoder:

$$E_u = \{eu_{u_1}, eu_{u_2}, \ldots, eu_{u_m}\} = WEncoder(u_1, u_2, \ldots, u_m)$$

where $u_j = \{w_{1j}, \ldots, w_{nj}\}$ is the $j$-th utterance.

2) **Utterance-level Encoder**: In a multi-turn dialogue, topics of different granularity are usually threaded through to drive the conversation. In addition, different speakers with different dialogue rhythms both play a dominant or passive role in the process of dialogue. Therefore, the topic and speaker serve as crucial contextual information. The utterance-level encoder in our model inspired by Transformer [19] focus
on context-aware modeling, processing all word-level encoded utterance in the dialogue history from the perspective of the topic and speaker to obtain the semantic information among utterances. To be specific, $S_{dt}, S_{gt}, S_{sp}$ as fixed-length vectors with the same length as the dialogue history, are fed into utterance-level encoder with sentence embedding. Then, in addition to the two granularity of topics, in order to further capture the relationship between the topic and the speaker, we introduce the heterogeneous multi-head self-attention with three kinds of masks, which can be formulated as:

$$\text{Attention}(Q, K, V, M) = \text{softmax}_\text{seq} \left( \frac{QK^T}{\sqrt{d_k}} + M \right) V \quad (2)$$

where $M$ is the mask matrix, determines whether the utterance can be attended to. Specifically, according to $S_{dt}, S_{gt}, S_{sp}$, we introduce several segmentation masks $M_{\text{type}}$ as shown in Fig. 2(c), which can be defined as:

$$M_{ij}^{dt} = \begin{cases} 0, & \text{same dynamic topic} \\ -\infty, & \text{otherwise} \end{cases}$$

$$M_{ij}^{gt} = \begin{cases} 0, & \text{same general topic} \\ -\infty, & \text{otherwise} \end{cases}$$

$$M_{ij}^{sp} = \begin{cases} 0, & \text{same speaker} \\ -\infty, & \text{otherwise} \end{cases}$$

where $i, j$ is the position of the utterance in the dialogue. Each head in multi-head self-attention mechanism is defined as:

$$h_i = \text{Attention} \left( E_uW_iQ, E_uW_iK, E_uW_iV, M \right) \quad (3)$$

where $W_iQ \in \mathbb{R}^{d_{\text{model}} \times d_q}, W_iK \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_iV \in \mathbb{R}^{d_{\text{model}} \times d_v}$ are learnable parameter matrices, $d_q, d_k, d_v$ denote the dimension of Query vectors, Key vectors and Value vectors. Each head focuses on different contextual information and the multiple contextual information is fused through concatenation, which can be formulated as:

$$E^c = \text{UEncoder}(eu_{t_1}, eu_{t_2}, \ldots, eu_{t_m})$$

$$= \text{Concat} \left( \text{Head}_{\text{base}}, \text{Head}_{\text{segt}} \right) W^O \quad (4)$$

where $W^O \in \mathbb{R}^{hd_x \times d_{\text{model}}}, h$ denotes the total number of heads and $x + y = h$. Head$_{\text{base}}$ denotes the set of heads without segmentation masks and Head$_{\text{segt}}$ denotes the set of heads with three kinds of segmentation masks. $E^c$ denotes the context representation.

3) Decoder: The response is generated by a recurrent decoder token by token with attention.

$$h_{t}^{\text{dec}} = \text{BiGRU} \left( h_{t-1}^{\text{dec}}, e_{t-1}; C_t, S_t \right)$$

$$C_t = \text{attention}(h_{t-1}^{\text{dec}}, E^c)$$

$$S_t = \text{attention}(h_{t-1}^{\text{dec}}, E^s) \quad (5)$$

where $h_{t}^{\text{dec}}$ denotes the hidden state at the $t$ step, $e_{t-1}$ denotes the word embedding of a generated word at the $t-1$ step, $C_t$ and $S_t$ denotes the representation of context and the same speaker as the respondent with additive attention weight.

$$y_{t+1} = \text{Softmax}(W_{\text{dec}}h_t^{\text{dec}}) \quad (6)$$

where $y_{t+1}$ denotes the generated word, $W_{\text{dec}}$ is a matrix that aligns $h_{t}^{\text{dec}}$ dimension with the dimension of the target vocabulary.

IV. EXPERIMENTS

A. Datasets

We evaluate our model on the following open-domain datasets:

DailyDialog [21] is a high-quality, multi-turn dialog dataset, the manually labeled dialogs of which are human-written to reflect our daily conversations.

Cornell Movie Dialog Corpus [22] is a dialog dataset of fictional conversations extracted from raw movie scripts. For each dataset, we split the corpus into the training set, validation set and test set at the ratio of 8:1:1, and reserve the dialogue containing more than 3 utterances. More details of the two datasets are shown in TABLE I.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DailyDialog</th>
<th>Cornell Movie Dialog Corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td>dialogs number</td>
<td>64190</td>
<td>93513</td>
</tr>
<tr>
<td>vocabulary</td>
<td>18091</td>
<td>42315</td>
</tr>
<tr>
<td>turn length</td>
<td>6.52</td>
<td>6.21</td>
</tr>
<tr>
<td>utterance length</td>
<td>14.54</td>
<td>11.62</td>
</tr>
</tbody>
</table>

B. Implementation Details

Our model is implemented using Pytorch. For topic segmentation, we load the pre-trained "all-mpnet-base-v2" for Sentence-BERT to get representations of each utterance. At the stage of topic segmentation, for obtaining dynamic topic boundary, we set the max length of the text block 7; for obtaining general topic boundary, we set the max number of the text block 3. At the stage of response generation, the numbers of hidden nodes are all set to 300, and the encoder and decoder layers are set to 10, 8. We set the number of heads in the multi-head mechanism of the utterance-level encoder as 12 and the ratio of the $head_{\text{base}}, head_{\text{topic}}, head_{\text{topic}}$ and $head_{\text{speaker}}$ are $4:2:1:1$. The decoding strategy we use is top-k sampling and nucleus sampling. During training, we set the batch sizes to 32 and 16 for DailyDialog and Cornell Movie Dialog Corpus datasets, respectively. The initial learning rate we set to 0.0001. Adam is used as our optimizer. We train our models at least 100 epochs on RTX 2080Ti Gpus.

C. Baselines

We compare our model with the following baselines with end-to-end framework on response generation task:

Seq2Seq [23] is a model with attention mechanism uses the sequence-to-sequence framework.
TABLE II
EXPERIMENTAL RESULTS ON DAILYDIALOG AND CORNELL MOVIE DIALOG CORPUS WITH AUTOMATIC EVALUATIONS.

<table>
<thead>
<tr>
<th>Model</th>
<th>Average</th>
<th>Extrema</th>
<th>Greedy</th>
<th>Distinct-1</th>
<th>Distinct-2</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>0.596</td>
<td>0.759</td>
<td>0.488</td>
<td>0.014</td>
<td>0.069</td>
<td>0.212</td>
</tr>
<tr>
<td>VHRED</td>
<td>0.611</td>
<td>0.773</td>
<td>0.497</td>
<td>0.028</td>
<td>0.129</td>
<td>0.214</td>
</tr>
<tr>
<td>HRAN</td>
<td>0.634</td>
<td>0.783</td>
<td>0.527</td>
<td>0.027</td>
<td>0.158</td>
<td>0.238</td>
</tr>
<tr>
<td>ReCoSa</td>
<td>0.615</td>
<td>0.770</td>
<td>0.497</td>
<td>0.031</td>
<td>0.157</td>
<td>0.216</td>
</tr>
<tr>
<td>TSHED</td>
<td>0.641</td>
<td>0.798</td>
<td>0.523</td>
<td>0.029</td>
<td>0.104</td>
<td>0.339</td>
</tr>
</tbody>
</table>

Dataset: DailyDialog

<table>
<thead>
<tr>
<th>Model</th>
<th>Average</th>
<th>Extrema</th>
<th>Greedy</th>
<th>Distinct-1</th>
<th>Distinct-2</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>0.529</td>
<td>0.625</td>
<td>0.439</td>
<td>0.006</td>
<td>0.030</td>
<td>0.121</td>
</tr>
<tr>
<td>VHRED</td>
<td>0.531</td>
<td>0.632</td>
<td>0.442</td>
<td>0.008</td>
<td>0.039</td>
<td>0.126</td>
</tr>
<tr>
<td>HRAN</td>
<td>0.522</td>
<td>0.623</td>
<td>0.438</td>
<td>0.007</td>
<td>0.049</td>
<td>0.127</td>
</tr>
<tr>
<td>ReCoSa</td>
<td>0.521</td>
<td>0.629</td>
<td>0.436</td>
<td>0.007</td>
<td>0.032</td>
<td>0.121</td>
</tr>
<tr>
<td>TSHED</td>
<td>0.548</td>
<td>0.633</td>
<td>0.455</td>
<td>0.006</td>
<td>0.052</td>
<td>0.129</td>
</tr>
</tbody>
</table>

Dataset: Cornell Movie Dialog Corpus

VHRED [5] is a HRED-based model combined a latent variable into generation process.

HRAN [6] is a hierarchical attention framework model focusing on the important information in context.

ReCoSa [13] is a dialogue generative model which makes full use of self-attention mechanism to find the relevant contexts.

D. Metrics

We adopt the following automatic metrics to evaluate the generated response:

Average, Extrema, Greedy [5] are the embedding-based metrics to measure the semantic similarity to the ground-truth response. We use the pre-trained Word2Vec word embeddings on the Google News Corpus for evaluation.

Distinct-n [3] is the metrics for reporting the degree of diversity, which is defined as the ratio of unique uni/bigrams to the total number of uni/bigrams in generated responses.

BERTScore [24] is the diversity metric used in the task of text generation, which computes a similarity score between two sentences that uses pre-trained BERT feature extraction.

E. Results and Analysis

TABLE II shows the results on two datasets. Generally, the proposed TSHED outperforms other models in most metrics on both datasets. As a unique non-hierarchical architecture in the experiment, Seq2Seq performs the worst. As the attention-based models, HRAN, ReCoSa, TSHED outperform VHRED on DailyDialog dataset. But on Cornell Movie Dialog Corpus dataset, only TSHED outperform VHRED. One reason for the result is that the dialogue turn and utterance length of DailyDialog dataset are longer than that of Cornell Movie Dialog Corpus dataset. The carefully designed attention mechanism in models can be fully utilized. Cornell Movie Dialog Corpus dataset is small in size of a dialogue, but very diverse and complex in content and style [25]. In addition, Cornell Movie Dialog Corpus dataset contains conversations from movies, and the character feature is more distinct, so the speaker-aware TSHED outperforms.

F. Ablation Study

In order to better understand the impact of the different context-aware attention mechanisms in our model, we have conducted experiments with different attention-based utterance-level encoders on the datasets. We denote TSHED without topic-aware mask as THED, TSHED without speaker-aware mask as THED. As shown in Fig.3, the performance of THED and SHED is different on different datasets, but TSHED achieves better performance than TSHED, demonstrating that all of our designed masks which is related to information flow in a dialogue are critical to context modeling.

G. Case Study

To better illustrate the coherent response TSHED generates in multi-turn dialogue, we provide a long multi-turn(12 turns) dialogues in TABLE III from the DailyDialog dataset. Seq2Seq, VHRED, HRAN tend to generate general and emotional responses which are not in line with the background of business negotiation. Compared with the response of ReCoSa, that of TSHED is more coherent and human-like. Note that the longer a dialogue, the more information about topics, scenes and speakers. From the case above, we find that with the help of explicit prompt in context, responses generated in multi-turn dialogues are not only relevant but also meaningful.
V. CONCLUSION AND FUTURE WORK

In this paper, we propose a topic and speaker-aware hierarchical encoder-decoder (TSHED) model to enrich informative context for response generation in multi-turn conversations. We apply a two-stage strategy to obtain topic boundaries and then adopt a hierarchical framework to achieve context-aware modeling. Experimental results show that our model performs well on DailyDialog and Cornell Movie Dialog Corpus datasets.

In future work, we plan to improve the effect of downstream tasks in multi-turn dialogue by combining the two stages of topic segmentation and dialogue context encoding or introducing emotion detection into word-level encoder.

REFERENCES


Convolution Neural Network Based Patent Infringement Detection Method

Weidong Liu
College of Computer Science, Inner Mongolia University,
Hohhot, China
cslwd@imu.edu.cn

Senjun Pei
College of Computer Science, Inner Mongolia University,
China.
Hohhot, China
peisenjun@gmail.com

Abstract—With the development of intellectual property rights in recent years, the number of patent applications has been increasing. At the same time, the number of patent infringement cases has also increased. When there is infringement between patents, the traditional method is for patent examiners to manually search for infringing features to determine whether there is infringement between patents according to the patent law. Since a patent is a complex semi-structured text and involves a wide range of fields, most of the current infringement detection methods cannot determine the infringement features well, and most of the methods only study one-to-one patent infringement and do not solve the problem of one-to-many patent infringement well. In order to solve the above problems, a patent infringement detection method based on convolutional neural network is proposed. The method extracts and represents infringement features from patents, patent claims and independent patent claims respectively, represents patents by different patent text vectorization methods, combines and filters features based on convolutional neural networks so as to obtain semantic information of different abstraction layers of patents, and finally tests the evaluation model on a one-to-many patent infringement data set. The results show that the model has greatly improved the infringement detection accuracy.

Keywords-CNN; Patent infringement; Infringement detection; Feature representation

I. INTRODUCTION

With the advent of the era of knowledge competition, the number of patent applications as a carrier of intellectual property rights has increased dramatically, leading to a high incidence of various patent infringement cases at the same time. Since patents are complex semi-structured texts, they cover a wide range of fields. When patent infringement occurs, traditional manual detection requires patent examiners to have a high professional background and need to spend time to understand and learn the relevant patent field knowledge, which not only increases the workload of manual detection, but also cannot guarantee the accuracy and timeliness of examination results, resulting in the accumulation of patent applications and unnecessary patent litigation, bringing economic losses to the patentee. Therefore, from the patent infringement cases, we can see the necessity and importance of designing a method for automatic detection of patent infringement.

In the current patent infringement determination process, the related work of infringement detection methods can be divided into supervised methods and unsupervised methods.

1) The unsupervised methods are broadly divided into clustering-based methods [1], which mostly use the k-means algorithm for clustering categories that are clear, and the SOM (Self-Organizing Maps)-based Chinese patent infringement detection algorithm [2] for clustering with fuzzy categories; game-theory-based methods [3], which are based on the claims of both sides of the patent and The method is based on game theory, which establishes a game tree based on the dynamic simulation of the claims and technical features of both parties to the patent, so as to calculate the risk of infringement between patents; based on the content of the patent text for keyword extraction [4], text content analysis to extract the SAO structure [5] to calculate its similarity, and determine whether the patent is infringed according to the size of the similarity; during the litigation processing of patent infringement cases, the patentee establishes a two-segment game based on the cost and profit of the patent to achieve Dynamic negotiation [6] to determine whether the patent is infringed and the post-infringement treatment.

2) Supervised methods are broadly classified into vector space model-based detection methods [7-8], which determine the negotiation relationship between patents by mapping all texts, paragraphs or words of patents into vectors of fixed size and calculating the similarity between vectors; hierarchical keyword vector construction based on patent claims [9], which calculates the patent-claim-technical feature hierarchical vectors by constructing the similarity between different patents and determine whether they infringe based on the magnitude of the similarity; based on the similarity between patents and considering both semantic and technical similarity, the method uses BERT to measure the semantic similarity based on the patent text [10]. Deep learning-based method [11-12], which extracts parts of patents and vectorizes them based on SOM neural network, and then clusters them using FCM algorithm; based on Doc2vec vectorization to judge the similarity between patent texts [13], which preprocesses patent texts and constructs a corpus, then vectorizes patent texts...
based on Doc2vec model. Then, the cosine similarity between vectors is calculated, and the size of similarity is proportional to the risk of infringement [14].

However, current infringement detection methods have the following limitations.

1) For unsupervised methods, the limitation of clustering-based methods is that the accuracy of clustering is low when solving large scale patent data sets, and the number of initial clusters cannot be determined, resulting in the k-means method cannot be used and the clustering effect is poor. The method based on game theory can only calculate the infringement risk value among patents to a certain extent, and the feature extraction single does not make good use of the patent text content to extract the features that represent patents. Similarity-based methods, because of the variability of patent text content and structure, increase the cost of manual labeling makes patent feature extraction more limited, and a large number of traditional similarity calculation only extracts part of the patent text information, and does not obtain the deep semantic information of the patent text, thus making the calculation of similarity more complex. The dynamic negotiation based on cost and profit is affected by the uncertainty of the negotiation process, and the result is contingent and cannot be efficiently and accurately determined whether the infringement has occurred.

2) For the supervised method, the vector space model-based method, the complexity of the patent text content makes the corresponding patent vector dimension and the semantic granularity of the patent vector features increased uncertainty, and most of the vector dimension is high and sparse, which is not conducive to the calculation of similarity. The deep learning method Yoon uses self-organizing neural network method to construct feature maps. Deep learning models require a large number of data sets, and the pre-model data processing and labeling work requires a lot of manual participation. The method of using Doc2vec to determine patent infringement is implemented based on text similarity. The method firstly represents the patent text content by embedded vectors with trained neural network models, and then calculates the similarity between the vectors, so as to determine whether the patent constitutes infringement between them based on the similarity magnitude. The limitation of this method is that it has a single feature extraction, does not integrate the patent text content well, cannot represent the patent text content efficiently and accurately, and has a low accuracy rate.

3) Most of the current infringement detection methods deal with one-to-one infringement relationships, and there is no fast and efficient solution for one-to-many patent infringement. Moreover, most of the traditional infringement detection methods are based on the textual content of patents, the cross-referencing relationship of patents and the non-textual content of patents (patent numbers, patent classification codes, application dates, etc.), and do not focus on the overall content of patents. Moreover, the existing detection methods do not have unified high-quality data sets, and the experimental results and models cannot be judged by their merits.

In order to determine whether multiple patents infringe each other, we propose a convolution neural network-based algorithm that uses the neural network to self-learn in order to discover patterns and obtain an efficient detection model.

1) A high-quality one-to-many patent infringement data set was constructed to better assess the strengths and weaknesses of the model and its generalization capabilities.

2) Extract features by fusing different parts of the patents. Although traditional patent infringement determination is based on the claims, other elements of the patent can also be used to help determine whether a patent is infringed.


II. BACKGROUND AND RELATED WORK

A. Doc2vec

One of the main steps of text classification is the word vector representation of text. A good word vector can better express the semantics between words. Word2vec is widely used in the field of word embedding. Although word2vec provides high-quality word vectors for sentences, documents or paragraphs, these data cannot be well projected into the vector space, nor can it express their rich semantic information. Doc2vec method is an unsupervised algorithm, which can learn from variable length text and obtain fixed length feature representation. Doc2vec model is an extension of word2vec model, and it also has two training methods, PV-DM (Paragraph Vector-Distributed Memory) and PV-DBOW (Paragraph Vector-Distributed Bag of Words).

B. Convolution Neural Network

Convolution neural network is a variant of feed forward neural network. Convolution layer is the core of convolution neural network. Through convolution operation, the two purposes of dimension reduction and feature extraction can be achieved. Convolution operation can select more representative local features, so as to express the important features of data more efficiently. At first, convolution neural network made a breakthrough in the field of computer vision [15], because it shows a high degree of displacement invariance in feature extraction. The two main features of local perception and weight sharing make the neural network effectively reduce the order of magnitude of parameter learning. When this idea is applied to text data mining, it greatly improves the performance of the task.

C. Patent Representation

A patent is a complex structure and a very broad field of public documents. Patent documents contain the title, specification abstract, claims and additional description of the patent, and as the core of the patent, the claims are essentially a collection of technical features of the patent. According to the description of the patent content, the claims are divided into independent claims and dependent claims. The independent claims contain the necessary technical features of the patent, and the dependent claims are attached to the independent claims, which are more detailed definitions and descriptions of the technical features of the independent claims. For patent infringement, the focus is not only on the identification of infringement relationship and infringement
relevance mining, but also on the consideration of the complexity of the patent content structure is particularly important for patent text mining [16-17].

**Definition 1: Patent Representation** \( P_i \)

\[
P_i = \left\{ A \cup C_{all}^{i} \cup D, C_{all}^{i}, C_{ij}^{i} \right\}
\]  

(1)

Where \( P_i \) denotes the \( i \)th patent, \( A \) denotes the abstract of the specification of the patent, \( C_{all}^{i} \) denotes all claims of the \( i \)th patent, \( D \) denotes an additional description of the patent, \( i \) denotes the number of all claims of the \( i \)th patent, \( C_{ij}^{i} \) denotes the \( j \)th claim of the \( i \)th patent, \( C_{ij}^{i} (C_{ij}^{i} \in C_{ij}^{i}) \) denotes the \( i \)th claim of the \( j \)th patent, and the claim is an independent claim. \( 0 < j < |C_{all}^{i}|, \sum_{j=1}^{k} C_{ij}^{i} = C_{all}^{i} \)

**Definition 2: Infringing patent data set IPDS**

\[
\text{IPDS= \{IPA}^{k}\}
\]

(2)

Infringing patent association \( \text{IPA}^{k} \)

\[
\text{IPA}^{k}=\left\{P_{i}^{0}, P_{i}^{1}, P_{i}^{2}, \ldots, P_{i}^{n}\right\}
\]

(3)

\( \text{IPA}^{k} \) represents a association of infringing patents analyzed through a set of real infringement cases, consisting of one infringing patent \( P_{i}^{0} \) and many infringed patents \( P_{i}^{1}, P_{i}^{2}, \ldots, P_{i}^{n} \). \( k \) represents the \( k \)th infringing patent association with infringing relationship in IPDS, an IPDS consists of \( n \) IPA.

**TABLE I. SYMBOLS AND THEIR MEANINGS**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_i )</td>
<td>The ( i )th patent</td>
</tr>
<tr>
<td>( A )</td>
<td>Abstract of patent specification</td>
</tr>
<tr>
<td>( C_{all}^{i})</td>
<td>All claims of the ( i )th patent</td>
</tr>
<tr>
<td>(</td>
<td>C_{all}^{i}</td>
</tr>
<tr>
<td>( D )</td>
<td>Additional description of the patent</td>
</tr>
<tr>
<td>( C_{ij}^{i} )</td>
<td>The ( i )th Claim in the ( j )th patent</td>
</tr>
<tr>
<td>( C_{ij}^{i} )</td>
<td>The ( i )th Claim in the ( j )th patent, which is an independent claim</td>
</tr>
<tr>
<td>( \text{IPA}^{k} )</td>
<td>Infringing patent association</td>
</tr>
<tr>
<td>IPDS</td>
<td>Patent infringement data set</td>
</tr>
<tr>
<td>( P_{i}^{k} )</td>
<td>The ( k )th one-to-many infringing patent</td>
</tr>
<tr>
<td>( P_{i}^{k} )</td>
<td>The ( k )th one-to-many infringing patent</td>
</tr>
</tbody>
</table>

**D. Problem Statement**

Patent infringement is the use of patented technology in production without the permission of the patentee or legal protection during the validity of the patent. The essence of one-to-many patent infringement is that when a set of technical solutions wants to be registered as a patent, it is necessary to determine whether the registered patented technical solution has been copied based on the technical features of the technical solution (evidence of infringement).

The paper is to address the patent infringement associations \( \text{IPA}^{k}, P_{i}^{0} \) whether the patent is infringed \( P_{i}^{1}, P_{i}^{2}, \ldots, P_{i}^{n} \). To solve this problem, we parse large-scale patent infringement cases and patent texts, extract different parts of the patent texts by parsing them, and denote them as patent word vectors by advance training patent text vectorization methods, and input the word vectors into a convolution neural network model through certain calculations, the convolution neural network includes multiple convolution layers, maximum pooling layers and fully connected layers. decay regularization. An excellent patent infringement detection model is obtained after training. The model can make a determination of whether a single patent infringes other patents.

**III. PROPOSED MODEL**

When one-to-many patent infringement occurs, infringing patent data set(IPDS) and infringing patent association (IPA\(^k\)) can be obtained by parsing the patent infringement cases.First the data set is divided into training set: validation set: test set as 6:2:2,and then parsed the patent text,obtain three representations of the patent: ① all patent contents \( A \cup C_{all}^{i} \cup D \), ② all claims of the patent \( C_{all}^{i} \), ③ independent claim \( C_{ij}^{i} \). The adjudication process is shown in Fig.1.

We train Doc2vec with the content of ① to get our vectorization method Patent2vec, and train Doc2vec with the content of ② to get our vectorization method Claims2vec.The patented text content needs to be converted into a vector representation by word embedding, different vectorization methods have a significant impact on the accuracy of text classification. Therefore, we train two different vectorization methods by analyzing the three patent contents obtained from the above patents. When the training samples of the input word or sentence vector Paragraph Id of the input word are from representation①, this method is Patent2vec, and when the training samples of the input word or sentence vector Paragraph Id of the input word are from representation②, this method is Claims2vec.

As shown in Equation 1, the first patent representation method uses the entire text content of the patent for representation, and the content of the patent text is vectorized with Patent 2 vec, and then trained with the constructed...
convolutional neural network, whose algorithm is shown in Algorithm 1.

Algorithm 1: Patent representation for patent infringement detection

Input: IPDS = \{IPA^k\}, IPA^k = \{p_0^k, p_1^k, p_2^k, ..., p_j^k\}

Output: Infringement result 0/1
1: Express all text contents of the patent as patent features, \(P_i = \langle A \cup C_{i all} \cup D \rangle\)
2: Vectorized representation of patent acquisition, \(V_{P_i} = \text{Patent2vec} (P_i)\)
3: for \(k = 1 \ldots K\) do
4: for \(j = 1 \ldots J\) do
5: Cosine_Similarity: \(W_l = \{p_i^k | p_j^k\}\)
6: Normalization \(W_i\)
7: \(V = V_{P_0} + \sum_{i=1}^J W_i \ast V_{P_1}\)
8: End
9: \(V_{input} = V\)
10: End
11: \(V_{input}\) Input to neural network prediction, return results

The patent text is analyzed to obtain a second representation of the patent, using the independent claims of the patent \(C_{i all}\) denotes the patent, and the content of the patent text is vectorized using Claims2vec and then trained using the constructed convolution neural network, it is shown as Algorithm 2.

Algorithm 2: Patent representation for patent infringement detection

Input: IPDS = \{IPA^k\}, IPA^k = \{p_0^k, p_1^k, p_2^k, ..., p_j^k\}

Output: Infringement result 0/1
1: Express all text contents of the patent as patent features, \(P_i = C_{i all}\)
2: Vectorized representation of patent acquisition, \(V_{P_i} = \text{Claims2vec} (P_i)\)
3: for \(k = 1 \ldots K\) do
4: for \(j = 1 \ldots J\) do
5: Cosine_Similarity: \(W_l = \{p_i^k | p_j^k\}\)
6: Normalization \(W_i\)
7: \(V = V_{P_0} + \sum_{i=1}^J W_i \ast V_{P_1}\)
8: End
9: \(V_{input} = V\)
10: End
11: \(V_{input}\) Input to neural network prediction, return results

The patent text is analyzed to obtain a third representation of the patent, using the independent claims of the patent \(C_{i all}\) denotes the patent, and the content of the patent text is vectorized using Claims2vec and then trained using the constructed convolution neural network, it is shown as Algorithm 3.

Algorithm 3: Patent representation for patent infringement detection

Input: IPDS = \{IPA^k\}, IPA^k = \{p_0^k, p_1^k, p_2^k, ..., p_j^k\}

Output: Infringement result 0/1
1: Express all text contents of the patent as patent features, \(P_i = C_{i all}\)
2: Vectorized representation of patent acquisition, \(V_{P_i} = \text{Claims2vec} (P_i)\)
3: for \(k = 1 \ldots K\) do
4: for \(j = 1 \ldots J\) do
5: Regular and Analysis \(C_{i all}\)
6: Obtaining \(C_{i j}\)
7: End
8: \(V_{input} = C_{i j} + \sum_{i=1} W_{i\ast C_{i j}}\)
9: End
10: \(V_{input}\) Input to neural network prediction, return results

IV. EXPERIMENT

To validate the efficiency of our model, we design experiments as follows. A. Experimental Data

In order to prove the validity of our model in rating prediction, we used the data of USPTO and Google patents as experimental data sets. Patent infringement cases are extracted from the patent trial and appeal board (ptab) inter departmental review (IPR) documents on the website of the U.S. patent and Trademark Office through automatic crawlers. These data sets include 100 patent infringement cases, 251 patent infringement combinations, and 2000 unrelated or similar patents. The patents used in our experiment are shown in Table II.

<table>
<thead>
<tr>
<th>Source</th>
<th>USPTO</th>
<th>USPTO</th>
<th>Google patents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>Patent infringement cases</td>
<td>IPA</td>
<td>Irrelevant or similar patent</td>
</tr>
<tr>
<td>Number</td>
<td>100</td>
<td>251</td>
<td>2000</td>
</tr>
<tr>
<td>Does infringe</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

B. Baseline Models

We compare our approach with state-of-the-art methods: a comparative study of Doc 2 vec based methods for detecting similarity in patent documents [11], which includes the following steps.

1) The patent is decomposed into a patent number document B and a patent abstract document C. The dictionary document set D is generated from document C, and D is used to generate the trained document set H using Doc2vec.
2) The similarity between the B and H patent number files is calculated and a manual secondary analysis is performed to obtain the final infringement detection results.

The essence of the method is to vectorize the patent text, and determine whether there is an infringement relationship between patents by calculating the similarity of the patent text. The size of the similarity is positively related to the probability of infringement.

C. Evaluation Measurements

Infringement risk is the evaluation measure in this paper. We will compare the experimental results of our proposed approach with the baseline approach in terms of violation risk. In this study, the patent associations violation problem is essentially a classification task. For the three sets of comparison experiments, we use score in Eq. 4 as the evaluation metric on the test set to judge the three sets of experiments, we use the most basic evaluation metric, precision in Eq. 5, to validate the predictions, we use a loss function to check the robustness of our constructed model, and we use F1 scores to balance precision and recall in Eq. 6 and in Eq. 7.

\[
\text{Precision} = \frac{\text{correct predictions}}{\text{all predictions}} \quad (4)
\]

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \quad (6)
\]

\[
F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Precision} + \text{Recall}} \quad (7)
\]

D. Experimental Setups

For the infringing patent associations, we conduct the following experiments.

1) Firstly, we obtain the judgment documents of patent infringement from USPTO, intercept and analyze the content of the judgment documents, obtain the infringing patent pairs and obtain the text content of corresponding patents from Google Patents and similar patents without patents to build the experimental data set. Parsing the patent text, extracting all the contents of the patent text, claims and independent right contents, i.e., Equation 1. generating training corpus to train Doc2vec to get vectorized methods Patent2vec and Claims2vec.

2) Based on the different representations of patents, the infringement detection model based on convolutional neural network is trained by Algorithms 1, 2 and 3, and the infringement features that work best in the infringement determination process are obtained by experimentally comparing the different representations of patent texts. When Patent2vec and Claims2vec are used to integrate features and statements, an excellent infringement detection model is finally obtained through experiments.

3) The Baseline Models is used for infringement detection through a comparative study of Doc2Vec based patent document similarity detection methods. Baseline Models Use IPA to verification. According to the vectorization method Doc2Vec in the Baseline Models, vector the different patent, and the infringement probability value between patents, namely the infringement risk, is calculated by using the similarity.

4) We conducted experiments based on the same data set IPA, then compare the results obtained by our method with those of the reference method in the evaluation measures.

E. Experimental Results

For the infringing patent data set, the data set is divided into training set: test set as 6:2:2. After data set partitioning is complete, our experimental flow is shown in Figure 1, where we represent the data by different patent representations and different patent text vectorization methods. The model was trained using the training and validation data, and once the training model was generated, the model was tested using the test data. Based on the evaluation parameters, our model was validated on the test data, as shown in Table III.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Loss(%)</th>
<th>Accuracy(%)</th>
<th>F1(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm1</td>
<td>0.6858</td>
<td>0.6000</td>
<td>0.74</td>
</tr>
<tr>
<td>Algorithm2</td>
<td>0.5638</td>
<td>0.7667</td>
<td>0.64</td>
</tr>
<tr>
<td>Algorithm3</td>
<td>0.5390</td>
<td>0.6667</td>
<td>0.73</td>
</tr>
</tbody>
</table>

In the Baseline Models, the patent number A to be tested and the abstract text of patent B to be tested are first generated from patent A in the post-training document H based on the trained Doc2vec model, and the list of similar patents is obtained and manually analyzed twice based on the abstract text of patent B. The basic experiment was performed by entering three patents for similarity determination, and the results are shown in Table IV.

<table>
<thead>
<tr>
<th>Patents similar to H</th>
<th>7874783</th>
<th>8995191</th>
<th>7999571</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similarity</td>
<td>0.5317</td>
<td>0.4655</td>
<td>0.4626</td>
</tr>
</tbody>
</table>

In the Baseline Models, the patent number A to be tested and the abstract text of patent B to be tested are first generated from patent A in the post-training document H based on the trained Doc2vec model, and the list of similar patents is obtained and manually analyzed twice based on the abstract text of patent B. The basic experiment was performed by entering three patents for similarity determination, and the results are shown in Table IV.

Compared with the benchmark method, the method proposed in this paper experiments with different manifestations of the patent and concludes through the experimental results that the best evidence of infringement is the claim part of the patent, because the claim part expresses the content of the patent clearly and in detail, and describes the
scope of protection of the patent in detail. Our method can quickly resolve the claims of patents when inputting one-to-many infringement cases and represent the claims by claim2vec, and finally get the result of whether the patent is infringed or not with an accuracy rate of 76.67%, while in the baseline model, when inputting infringing patents and infringed patents, the trained Doc2vec model is used to generate the patents into training. After the document, as the greater the similarity between infringing patents, the greater the risk of infringement between patents, so, although the baseline model solves the complex problem in the patent field to a certain extent, its accuracy rate is still low.

V. CONCLUSIONS AND FUTURE WORK

The essence of patent infringement detection is text classification. The key is to accurately extract the central idea of patent documents. The method of extracting the central idea is to extract the keywords of documents or sentences as features, and train classifiers and classify them based on these features. Because the convolution and pooling process of convolution neural network is a feature extraction process, when we can accurately extract the features of keywords, we can accurately extract the central idea of documents or sentences. The baseline model method only calculates the similarity based on the representation of patent documents, does not refine the content of patent documents, but uses mechanical methods to represent patents. It is impossible to accurately judge whether patents are infringed, but it has reference value to some extent. A patent infringement detection method based on convolutional neural network is proposed in this paper. The method obtains the depth information of the patent by selecting and representing the textual content of the patent, and merges different elements of the patent for comparison, so as to obtain an effective determination method. The contribution of our method to previous research is summarized as follows:

1) The lack of standardized and consistent datasets for previous infringement detection methods has led to the inability to evaluate the model, which we support by analyzing decided infringement cases from uspto.

2) Using an innovative and unified convolutional neural network automatic infringement detection framework, the model is trained with different patent text representation methods and text vectorization methods, and the best patent text representation method for infringement detection is obtained through comparative experiments, which improves the detection efficiency to some extent.

3) Most of the previous infringement detection focuses on the infringement detection of a single patent, and this paper solves the infringement determination of multiple patents.

We compare our approach with state-of-the-art methods and the results show that our approach outperforms the benchmark methods in patent infringement detection. In the case of one-to-many patent infringement, in addition to building a high quality dataset for patent infringement detection, better models can also be used to extract semantic information about patents. These issues will be addressed in future work.

ACKNOWLEDGEMENT

The research work in this paper was supported by the National Science Foundation of China (grant no. 62161036 and grant no. 61801251), Postdoctoral Science Foundation of China (2020M680643), Program for Young Talents of Science and Technology in Universities of Inner Mongolia Autonomous Region(NJYT23058).

REFERENCES


DDCL: A Dual Decision-making Continuous Reinforcement Learning Method Based on Sim2Real

Wenwen Xiao¹, Xinzhi Wang¹, Xiangfeng Luo¹,†, Shaorong Xie¹
¹School of Computer Engineering and Science
²Shanghai University, Shanghai, 200444, China
{xww, wxz2017, luoxf, srxie}@shu.edu.cn

Abstract—Continuous reinforcement learning carries potential security risks when applied in real-world scenarios, which could have significant societal implications. While its field of application is expanding, the majority of applications still remain confined to virtual environments. If only a single continuous learning method is applied to an unmanned system, it will still forget previously learned experiences, and retraining will be required when it encounters unknown environments. This reduces the learning efficiency of the unmanned system. To address these issues, some scholars have suggested prioritizing the experience playback pool and using transfer learning to apply previously learned strategies to new environments. However, these methods only alleviate the speed at which the unmanned system forgets its experiences and do not fundamentally solve the problem. Additionally, they cannot prevent dangerous actions and falling into local optima. Therefore, we propose a dual decision-making continuous learning method based on Simulation to Reality (Sim2Real). This method employs a knowledge body to eliminate the local optimal dilemma, and corrects bad strategies in a timely manner to ensure that the unmanned system makes the best decision every time. Our experimental results demonstrate that our method has a 30% higher success rate than other state-of-the-art methods, and the model transfer to real scenes is still highly effective.

KEYWORDS: Social Computing; Continuous learning; Reinforcement learning; Simulation to reality(Sim2Real).

I. INTRODUCTION

Continuous reinforcement learning plays a crucial role in our society by enabling the development of social software that enhances interactions within groups and improves the efficiency of human social activities [1] [2]. Continuous reinforcement learning involves acquiring skills through continuous interaction with a complex environment and building higher-level skills based on previously learned ones. The continuous reinforcement learning process is analogous to how babies learn to walk, where crawling is mastered first, followed by standing and eventually walking, as illustrated in Fig. 1. In essence, each new skill is built upon the old ones. However, there are still at least two challenges in continuous learning: avoiding catastrophic forgetting caused by neural networks and ensuring the malleability and stability of the models for migration to real-world scenarios.

The current approach to mitigate the forgetting problem of agents during the learning process is primarily through the priority experience replay mechanism [3]. However, this method requires frequent parameter adjustments and can result in reduced learning efficiency of the agent.

To ensure that the trained model of an agent in a virtual scene is steadily migrated to a real scene, the current mainstream approach uses adjusting while training to optimize the policy parameters of the agent in real time. However, this method requires a large amount of human resource cost and will become ineffective in the face of some complex scenarios.

In this paper, we propose a dual decision-making continuous learning method based on Simulation to Reality, which consists of three main stages: perception stage, decision stage, and execution stage.

In the perception stage, virtual data is first transformed into real data using existing data generation tools. Then, real data from the real scene is transformed into virtual data, and the resulting data is fused. Next, the fused data is passed through a semantic segmentation extractor to obtain a feature map, and object detection techniques are employed to extract entity category information in the scene. Finally, a semantic knowledge graph is constructed based on the feature map and entity category information, which serves as the agent’s prior knowledge during the decision stage. This prior knowledge can reduce the agent’s exploration of the scene and improve the efficiency of its decision-making.

In the decision stage, the agent’s decision-making action is controlled by a dual decision-making mechanism consisting of continuous learning and body-of-knowledge control methods. The agent learns different strategies through trial and error with various environments, with each strategy
forming a skill. To prevent forgetting of the learned skills, periodic updating of the training environment is necessary. Body-of-knowledge control is utilized to help the agent escape from local optimum in policy learning. The decision to apply body-of-knowledge control is based on the evaluation results of the discriminator, which consists of success rate, error rate, and reward value indicators.

In the execution stage, the agent’s actuators receive the optimal decision-making action determined by the discriminator evaluation to ensure compliance and safety, enabling the agent to quickly complete the task.

In summary, we propose a dual decision-making continuous learning method based on Simulation to Reality, which can effectively mitigate the problem of skill forgetting during the learning process of an agent. Moreover, this method is crucial for ensuring the stable migration of strategies learned by an agent in a virtual scene to a real scene.

The main contributions of our paper are summarized as follows:

1) We propose a dual decision-making continuous learning method based on Simulation to Reality, which effectively circumvents bad strategies and ensures the ability of continuous learning of the agent.

2) We improve the efficiency of the agent’s search for unknown environments by introducing semantic knowledge graphs as prior knowledge in the perception stage.

3) We migrated the trained model in the virtual scene to the real unmanned vehicle defensive scene, and the mobile vehicle can still complete the task smoothly.

II. RELATED WORK

A. Continuous reinforcement learning

Continuous reinforcement learning is a method to address catastrophic forgetting in neural network learning, which is critical in improving the efficiency of agents in social activities[4] [5]. This approach allows agents to learn different skills at different times, improving their continuous learning ability and avoiding the need to retrain the agent for new tasks. As a result, the agent’s learning efficiency is improved.

The mainstream methods are regularization, memory playback, parametric isolation, and integrated methods. Regularization methods are achieved by adding regular terms to the homeopathic function during training a new task and modifying the ratio of old and new data to reduce the rate of forgetting the agent. The memory replay method is to reuse the data that has been used before to reduce forgetting. The parameter isolation method is to assign different model parameters in different tasks of the agent training and freezes some model parameters in time according to the performance of the agent to ensure that the old model parameters occupy the majority[6]. The combined approach combines the above two approaches to form a new approach. For example, Buzzega combines regularization and memory replay to propose dark experience replay[7].

B. Mobile vehicle

Mobile vehicles have been applied across diverse industries, such as hospitals, factories, supermarkets, and hotels, profoundly impacting the social life of people today[8]. Among them, ground-guided vehicles stand out for their quick response time, fast speed, and high carrying capacity. However, most current mobile vehicles rely on rule-based control methods that are limited to simple scenarios[9] [10]. In complex scenarios, these methods exhibit poor performance and may even fail. Vision-based navigation and radar slam navigation are the two main rule-based control methods, both of which require environmental map information and suffer from low decision-making efficiency and poor migration[11].

With the rapid development of deep learning technology, continuous interaction between mobile vehicles and the environment via deep reinforcement learning has become a popular research direction for enabling autonomous decision-making[12]. To address the challenges of performance degradation and skill forgetting when trained models of mobile vehicles are migrated from virtual to real scenes, we propose a dual decision framework that successfully completes the red and blue vehicle defense task in real scenes with zero-shot transfer.

III. METHODS AND ANALYSIS

A. Dual Autonomy Decision Framework

The dual autonomous decision-making framework plays a very important role in improving the continuous learning ability of mobile vehicles and reducing the speed of skill forgetting, laying the foundation for large-avoidance swarm intelligence, as shown in Fig. 3. The framework is mainly divided into the perception stage, decision stage and execution stage, and the role of each stage is also different. In the perception stage, it focuses on how to obtain feature maps and reduce the state space of the ground mobile vehicle; In the decision stage, it focuses on how to improve the autonomous decision-making ability of the ground mobile vehicle; In the execution stage, it focuses on how to execute the output actions of the decision stage smoothly.

Perceptual Stage: The perception stage is the basis of the dual autonomous decision-making framework. The main
work of the perception stage has two parts, namely image generation and feature map construction, as shown in Fig. 2. Image generation uses a generative adversarial network to generate images in the virtual scene into images in the real scene, generate virtual scene images from the original images in the real scene, and then mix the virtual images and real images for training to obtain an image generation model.

**Decision Stage:** The decision stage is similar to the brain of the dual autonomous decision-making framework, providing continuous learning capabilities for mobile vehicles. The decision stage mainly consists of a policy network, a body of knowledge controllers and two discriminators. First, the mobile vehicle simultaneously learns multiple continuous learning policies, and each policy network is dependent on the critic and actor networks. Then use the evaluation index in discriminator 1 to evaluate the policy output in the policy network. If the evaluation result is not good, it will directly switch to the body of knowledge controller to correct its policy parameters. Finally, the optimal action is obtained through the evaluation conditions and indicators in discriminator 2.

**Execution Stage:** The main function of the execution stage is to ensure that the mobile vehicle can control the movement of the mobile vehicle smoothly according to continuous action. The execution stage comprises the underlying controller, PID module (Proportional, Integral, Differential), robot operating system (ROS) and API interface. Through the above modules, it can be seen that the torque of the underlying motor is converted into continuous speed and direction data, where the value range of speed and direction is -1 to between 1.

**B. Dual Decision Continuous learning algorithm (DDCL)**

We proposed the dual-decision continuous learning method based on the Proximal Policy Optimization (PPO) algorithm[13]. The pseudocode of DDCL algorithm is shown in algorithm 1. Because only relying on this method when facing some complex scenes, it is easy to forget the previously learned experience or fall into the problem of local optimum, which makes the mobile vehicle unable to perform continuous learning. Therefore, to solve the above problems, we also use the knowledge body control method to assist the decision-making of mobile vehicles. On the one hand, it can improve the efficiency of decision-making, and on the other hand, it can avoid the training of mobile vehicles from scratch. The policy network structure of this method is shown in Fig. 4. The network update method of the actor and critic depends on the PPO algorithm[13].

To ensure that the red car agent can learn defensive and patrol strategies, it is necessary to set the reward function skillfully. The reward function formula is as follows:

$$R_{total} = \begin{cases} +10 & \text{when } R_{car} \text{ is intercepted} \\ +5 & \text{when } R_{car} \text{ is on patrol} \\ -10 & \text{when } B_{car} \text{ is attacked the target} \end{cases}$$

where $R_{total}$ is the reward function for interacting with the environment.

According to the evaluation index in discrimination 1, the comprehensive strategy evaluation value $P_{total}$ is obtained,
and the optimal strategy is selected. The specific formula is as follows:

\[ P_{\text{total}} = W_1 \cdot P_e + W_2 \cdot P_r + W_3 \cdot P_w \]  

(2)

where Error rate \( P_e \) is represents the stationary time of the moving vehicle in each round; The risk rate \( P_r \) is represents the number of extreme actions of the moving vehicle in each round; The reward value \( P_w \) is represents the average reward value in each round.

According to the evaluation index in discrimination 2, the comprehensive action evaluation value \( A_{\text{total}} \) is obtained, and the optimal action is selected. The specific formula is as follows:

\[ A_{\text{total}} = \eta_1 \cdot A_g + \eta_2 \cdot A_s \]  

(3)

Where the action success rate \( A_g \) represents the number of times the vehicle swings in each round; The stability \( A_{ss} \) represents the offset of the moving vehicle from the centerline of the track in each round.

In summary, to prevent the mobile vehicle from forgetting the previously learned skills and improve the transferability of the model, it is necessary to adjust the strategy in time according to the actual situation of the comprehensive action and the evaluation value of the strategy. If the evaluation shows poor results, it’s necessary to switch directly to the knowledge-based control mode.

IV. EXPERMENTS

A. Scenario description and tools

Scenario description: The goal of the red and blue vehicle defensive task is to allow the red vehicle to intercept the blue vehicle in time through the double continuous learning decision-making algorithm to ensure the safety of the guard target. The main entities in the red and blue offensive and defensive scene have a red car, a blue car, and a target. Among them, the agent controls the red vehicle through the double continuous learning algorithm, the knowledge body controls the blue vehicle, and some fixed decision-making mechanisms are artificially set. The defensive interception scenes of the red and blue sides construct the same scene in virtual and real, respectively, assuming that the dynamic models of the moving vehicles in the virtual and real scenes are the same or similar.

Red and blue of vehicle defensive tasks: Blue vehicle decision-making mode: blue team vehicles drive at a constant speed in the outer lane of the scene and perform patrol tasks. It launches an attack on the target every 5s. If it is intercepted by a red vehicle, it will exit the attack mode and continue to execute the patrol mechanism. Red vehicle decision-making mode: The red vehicle pays close attention to the movement of the blue vehicle in real-time. If the blue vehicle has already driven to the blue inner circle, the red vehicle will start to intercept until the blue vehicle exits the inner blue area.

Scenario tools: We used the unity virtual engine [14] to create a virtual scene of the game between red and blue. The GPU in the server is an NVIDIA GeForce RTX3090 graphics card, and the CPU is Inter core i7-9700. The control driver of the mobile vehicle is ROS 18.04 LTS, and the controller is Jetson Xavier NX. The flow chart of real car model migration is shown in Fig. 9.

B. Unmanned vehicle train in virtual scenarios

Feature map construction: The feature map is to obtain the feature vector through the method of feature extraction from the original image, which can process the original high-dimensional image information into a low-dimensional feature vector, thereby improving the decision-making efficiency of the mobile vehicle. The feature map is obtained by data processing the images in the virtual and real scenes through the existing yolov5 method. The feature map consists

---

**Algorithm 1 Dual Decision Continuous learning**

1: **Input:** Initialize strategy, \( S_1, S_2, \ldots, S_i \). Initialize actor parameters: \( A_1, A_2, \ldots, A_i \). Initialize actor parameters: \( B_1, B_2, \ldots, B_i \). Initialize hyperparameters \( T_1 \) and \( T_2 \).
2: Choose a strategy at random.
3: **for** \( n \) **episode** **do**
4: **Update** Actor Network Parameters:
5: \( a_i \leftarrow a_i - T_1 \Delta a_i, N_{\text{Actor}}(a_i) \)
6: **Update** Critic Network Parameters:
7: \( b_i \leftarrow b_i - T_2 \Delta b_i, M_{\text{critic}}(b_i) \)
8: **end** **for**
9: Calculate the strategy evaluation value, see formula 2.
10: Calculate the action evaluation value, see formula 3.

**TABLE 1: Performance comparison in virtual scene**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Success Rate</th>
<th>Risk Rate</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>8%</td>
<td>92%</td>
<td>BAD +</td>
</tr>
<tr>
<td>SAC[15]</td>
<td>60%</td>
<td>40%</td>
<td>GOOD -</td>
</tr>
<tr>
<td>DDCL(Ours)</td>
<td>90%</td>
<td>10%</td>
<td>GOOD +</td>
</tr>
</tbody>
</table>

---

**Fig. 4: Policy Network Structure**
Decision Model Training: We connect the dual continuous learning algorithm to unity in the virtual engine for accelerated training. The overall picture of the red and blue of defensive vehicle tasks trained at different moments is shown in Fig. 5. Compared with the state-of-the-art algorithm, our proposed dual continuous learning algorithm has greater advantages in average reward value, error rate, success rate, risk rate and episode steps, as shown in Fig. 6 and Table I.

C. Unmanned vehicles verified in real scenarios

Migrate the model trained in the virtual scene to the real scene with zero-shot. The migration process is shown in Fig. 9.
in Fig. 9. It can be seen from Table II that the DDCL algorithm proposed by us has a success rate of 60% in real red and blue of defensive vehicle tasks, which is better than state-of-the-art algorithms. The risk rate reaches 40%, mainly caused by the kinematic differences of real vehicles, the ground’s friction coefficient and the light’s intensity. Fig. 10 shows the running trajectories of the red vehicle and the blue vehicle in the real scene of the mobile vehicle at different times. The view of the red vehicle is shown in the upper row of Fig. 11, and the view of the blue vehicle is shown in the lower row of Fig. 11.

V. CONCLUSIONS

The continuous reinforcement learning method in social computing is of great assistance in improving the efficiency of our social life. Therefore, we first developed a dual-decision framework to enhance the autonomous learning capability of mobile vehicles and ensure stable performance when they are applied in real-world scenarios. We then propose a dual decision-making continuous reinforcement learning method based on Simulation to Reality, which enables the mobile vehicle to avoid bad strategies and maintain continuous learning ability. Our experimental results demonstrate significant improvements in red and blue defensive vehicle tasks, and successful migration of the model to realistic scenarios with zero-shot. The mobile vehicle was able to complete the task smoothly. In the future, we aim to construct more complex game confrontation scenarios and introduce additional mobile vehicles to realize intelligent group games.

VI. ACKNOWLEDGMENT

The work reported in this paper was supported in part by the Ministry of Industry and Information Technology project of the Intelligent Ship Situation Awareness System under the grant No.MC-201920-X01, and National Natural Science Foundation of China under the grant No.61991415.

REFERENCES

Inclusive Gamification: An Exploratory Study in Software Development Enterprises

Fayrouz Elsalmy
Information Systems.
College of Computing and Information Technology
Arab Academy for Science and Technology
Alexandria, Egypt
fayrouz_elsalmy@aast.edu

Nada Hany Sherief
Software engineering.
College of Computing and Information Technology
Arab Academy for Science and Technology
Alexandria, Egypt
nada.sherief@aast.edu

Walid Abdelmoez
Software engineering.
College of Computing and Information Technology
Arab Academy for Science and Technology
Alexandria, Egypt
walid.abdelmoez@aast.edu

Abstract—Gamification is the incorporation of game elements into non-game settings. Software designers increase user motivation by introducing adequately engaging elements, such as leaderboards and badges, into an existing system. In recent years, few studies have examined the risks associated with gamification inside the business systems sector, especially within software development organizations. Yet, this issue must be addressed more systematically to build gamified solutions that take individual gamification risks into account. This research introduces "Inclusive Gamification" as contextually aware gamification that highlights individual risks. The purpose of this paper is twofold: 1) to identify the gamification risk factors that could cause the emergence of risks in software development organizations; and 2) to develop a list of risks with consideration of various contextual factors, namely tasks, personality type, adopted game element, and organizational cultural aspects. 15 employees from three software development companies participated in an exploratory study in which their perspectives on the negative effects of gamification were collected. As a result, we were able to identify several risk factors and associated risks. Our findings assist software engineers throughout the design process in identifying potential threats. Thus, create gamified solutions that better situate employees at their workplaces.

Keywords—Gamification; Software Development; Contextual Factors; Unexplored game elements; Gamification Risks; Risk Factors; Personality Types

I. INTRODUCTION

Deterding et al. define gamification as the "application of game element in non-gaming context" [1] to engage users and motivate them to achieve goals [1-3]. Gamification in Enterprise Systems (ES) has been the most intriguing and successful [3, 4]. We focus on gamification in software development enterprises within their internal software development life cycle phases, activities, roles, and tasks[5-9].

Gamification has been adopted in the field of software development to address challenges related to the software engineer’s motivation, performance, and engagement in their tasks [5, 10] such as bug fixing, freeing up the backlogs or writing unit testing, and documentation [6, 11, 12] but also related to achieving organizational objectives as developing higher product quality and project performance [5, 7, 8].

Gamification works best when personalized to its users [9, 13, 14]. However, using only one personalization aspect as the user type model to determine game element preferences is insufficient [15]. Most gamified software engineering studies use PBL (point, badges, and leader boards) as game elements [5, 8], omitting a broad group of game elements and using a “one size fits all” strategy, which defeats the core aim of gamification, which is adaptability and variety. Hence, [14-16] advised tailoring gamified solutions to game elements, environment, and users to improve user acceptance and system efficiency[13]. As for enterprise systems, factors such as user type [17], culture [5, 18-21], task [5, 17, 21], role[19], goals [19], motivational elements [22, 23], and organizational context [24, 25] were utilized. In our analysis, we resort to the GLOBE model's organizational culture dimensions[26].

Several studies examined gamification risks and risk variables for enterprise gamification [21, 27] and software (engineering) development organizations [4, 5, 7, 8] in particular. In [5] challenges included decreasing autonomy, people being cheated, demotivation, and decreased creativity. Yet, there was no additional emphasis or research on these challenges. That is, the primary causes for the emergence of the risk, when and how these challenges would occur, and the personalities who have encountered these risks. Hence, a significant gap exists, namely the inability to tie gamification risks to personal and contextual characteristics. Risks and risk factors for enterprise gamification systems in general and teamwork were presented[21]. They provided a checklist for identifying and mitigating risks. The risks were classified as either ethical and well-being, performance, or productivity risks.

In our study, we investigate the negative effect of other game elements that are more applicable to be used in software development enterprise systems, as previous studies [5] have neglected to consider who is susceptible to these risks and in what contexts, i.e., it is unknown which personality factors are more prone to experience such risks. This work investigates the interaction of four diverse contextual factors: 1) game elements

DOI Reference Number: 10.18293/SEKE23-096
with a particular emphasis on unexplored game elements 2) personality traits 3) organizational culture 4) the nature of the task, for a more tailored software development life cycle that accommodates all personality types and their motivators.

II. RESEARCH METHODOLOGY

In this research paper we introduce the term “Inclusive Gamification”. It is introduced as gamification with consideration of risk factors. This approach emphasizes equal considerations for all users in gamification design, including personalized risk identification. Four contextual factors, including game elements design, task nature, personality type, and organizational culture, are considered in the study. We followed a qualitative approach [28] due to the exploratory nature of the study, the interview approach was adopted to gather qualitative data in software engineering, providing in-depth insights from participants. Semi-structured interviews were chosen for their flexibility, allowing for core questions and exploration of relevant areas, and the opportunity to pursue ideas or responses in greater depth. In our study we strive to address two research questions: RQ1 what are the gamification risk factors that could cause the emergence of risks in software development organizations? And RQ2 what are the associated risks with consideration of various contextual factors, namely tasks, personality type, adopted game element, and organizational cultural aspects?

The study included 15 participants, consisting of 7 males and 8 females, aged between 25-34, from the USA and Egypt, with varying roles and seniority in software development companies. Participants had roles such as software engineers, software architects, project managers, product managers, quality assurance engineers, and UI/UX designers. Participants were recruited from three distinct software development companies that utilized gamified systems for managing internal processes and tasks. All three companies shared cultural values of innovation, openness, team loyalty, and cohesion, with a preference for group work and projects over individual work, indicating high collectivism. However, there were differences in cultural aspects such as power distance and uncertainty avoidance among these organizations.

Participants were recruited and interviewed individually. To ensure that we covered all the personality types prior to the interviews, we sent a personality traits quiz via email to each interviewee to answer the 44-item BFF questionnaire [29] to determine their personality type. We conducted 15 interviews, averaging 1.5 hours each, with one researcher conducting all sessions. Audio recordings and transcriptions were made and saved in the following link https://rb.gov/k81v37. Two pilot interviews were conducted to assess study viability.

The interview transcripts were analyzed using thematic analysis[30]. The results of the interview analysis, which are depicted in figure 1, helped us understand how to reach conclusions about the key contextual themes of the investigation, which are risk factors that are personal, organizational, task-related, performance-related, and game-element related. These are the primary causes of the rise of gamification risks in software development organizations.

III. RESULTS

Since we presented the term “Inclusive Gamification” in the previous section, we seek to identify the risks and clearly state the personal and contextual reasons behind them to extract and define how to develop inclusive gamification enterprise systems. Figure 1 addresses RQ1 and RQ2 is discussed in the rest of this section.

Figure 1 Thematic map for gamification risk factors in gamified software development enterprises

A. Organizational Related Risk Factors

The term organizational related factors encompass all influences on organizational behavior. Identifying and categorizing these factors is crucial in mitigating gamification risks, which may arise if organizational aspects like culture, management strategies, and structure are neglected. Participant interviews provided insights that allowed for conclusions to be drawn about organizational culture and its influential cultural dimensions, such as collectivism, power distance, and uncertainty avoidance. In a collectivism culture “I collect as much points and badges as I could just to appear on my team’s Leaderboard. We are competitive to do more, not to compete” leading to being self-centered.

The management style which was the manager’s style for planning, organizing, delegating the employees and how they announced and acknowledged the employees about the gamification rules and its adoption. Managers following authoritative management style in a gamified ES could lead employees to experience gamification risks such as effort-misinterpretation, “I earn points on x no of commits per day and changed x number of lines of codes on source control systems and get judged on them.”

The term organizational structure defines a strict chain of command within the organization. It describes who oversees what inside an organization and how its goals are to be met. In a flat and decentralized organization, “Being an agile project manager and working in a gamified environment, where earning and collecting points, badges and, Easter eggs, is one of my getaways during work to keep me engaged and immersed into work” leading to Procrastination, Addiction, and ruining work life balance.
B. Task Related risk Factor

The term task refers to a specific piece of work with clear parameters, and task factors encompass task type, assignment, and dependency, which are crucial aspects of an organization’s workflow involving employee task allocation. Task type can be categorized as Core or Non-core, with core tasks being essential to business success including typical and innovative tasks like bug fixing, data modeling, testing, creating wireframes, and required trainings. Non-Core tasks refer to routine tasks that do not directly contribute to the company’s success. Examples include fixing low priority bugs from backlog. In a typical core task, “I shouldn’t get rewards on my daily jobs as bug fixing. I could’ve solved more low priority bugs and get more rewards.” leading to lower task quality.

Task dependency refers to the relationship where the completion of tasks must occur in a certain order. “I sometimes overlook the quality of the outcome of the preceding task from my colleague in order to achieve my reward” leading to lower task quality.

Task assignment refers to the assignment of tasks to employees either explicitly by the project manager or implicitly by the employees themselves during the sprint. The choice of the task assignment depends on the team dynamics and team members being involved in the project. Each task is assigned to one individual to hold him/ her accountable for. Some employees are prone to experience gamification risks, when they are not being explicitly assigned by the project manager to do the task. “Progress bar assigned for the whole sprint is demotivating and is not accurate can lead to conflicts and problems between team members. Especially if I wasn’t assigned by the project manager to do the task” leading to Peer conflict & inaccurate achievement.

C. Performance Related risk factor

The term performance encompasses the efficiency and effectiveness of employees in carrying out assigned tasks, along with factors such as conduct, productivity, and overall value to the organization. Performance is monitored and assessed as a crucial aspect of evaluating employee performance at the workplace. Performance Feedback: refers to the feedback related factors such as feedback source, feedback comparison and the feedback timing and frequency. “Levels can underestimate my work. I could be working hard and efficient, yet this could not be reflected on the system” leading to feeling demotivated & stagnant.

Performance Assessment refers to the formal assessment used to assess the employee’s performance at their workplaces and the format of the feedback being adopted at the formal assessment. Due to the automated feedback from the system “I can easily cheat on the system and do some mechanical refactoring, renaming methods and file name with no impact at the end just to increase the commits number and increase the points I earn” leading to cheating to the system.

Performance Transparency refers to either the employees decide to be voluntarily transparent about their performance on communication channel amongst their team members during their task accomplishment or transparency within the company regarding their awareness and acknowledgment about the gamified platform being adopted. “On slack, others announce what tasks they finished. I for sure feel pressured if I haven’t finished my tasks” leading to increased pressure.

D. Personal related risk factor

This section defines the employee’s perception towards game elements and the emergence of gamification risks would differ according to some personal related factors. For example, personality type, seniority level, role and demographics are the personal related factors that we need to consider when designing inclusive gamified system. Participants were given the BFF 44-questionnaire prior to their interviews to determine their personality type.

Conscientiousness is motivated by either achievement elements as points, badges, leader boards, levels, progress bar and rewards or by gifting and sharing knowledge. “We have a leader board for showing our daily progress of the team, I set my targets to beat the best numbers of others” leading to Anchoring bias”. Agreeableness are motivated by social competition, teams/guilds, social network feature, knowledge sharing, and gifting. “In an optional training or fixing an optional task no need to be stressed and have a time pressure” leading to quitting task. Neuroticism Motivating elements: Easter eggs, customization, un-lockable content, voting mechanism and anonymity. “Competition at work shakes my confidence, it affects me mentally” leading to feeling unconfident. Openness personality type is motivated by immersion elements such as Easter eggs, un-lockable content, and avatars. “I want to help others just to get recognized Publicly on slack” leading to Ego-centric. Extraversion employees are motivated by immersion elements as Easter eggs, unlock able content and socially related elements as social competition, teams, social network feature and gifting and sharing knowledge. “I keep looking for the everyone’s achieved badges and Easter eggs keep trying to find them, I leave important tasks unachieved when the systems allow for such things to occur” would fall into a Rabbit hole trap (addiction).

Role & Seniority refers to the employee’s role (Software engineer, Designer, Project manager, etc.) and seniority level (junior, senior or an executive) at their workplaces and how they could be one of the reasons behind the emergence of gamification risks. “I would be stalling when I couldn’t reach higher levels in my training as other managers”, leading to managers being stalling & disengaged.

Demographics refers to the employee’s age and how would they affect the emergence of gamification risks. Employees with different age groups would respond to gamification differently. A misfit between the organization culture and demographics would result in gamification risks. “Overaged board members 50 + wouldn’t like to participate, they don’t have their gamified profiles accessible like us” leading to lack of participation.

E. Game element Related Risk factor

This section describes the game elements related factors as the reward type, the reward assignment level and the visibility and accessibility of the rewards.

Reward Type refers to the nature of the rewards being incorporated into the ES either tangible (bonus, vouchers),
virtual incentives (as avatars, points, leaderboards, and badges), social influences (competition, teams, gifting) and challenges. “I send kudos to my friends daily” leading to unfair judgment, clustering groups and Intimidation amongst employees.

Assigned to refer to the how rewards could be assigned to various levels either company, team, or individual level. “Labeling us with Levels on a company-level is unneeded” leading to jealousy and employee inequality.

Visibility & Accessibility: Visibility refers to featuring and displaying of the employee’s performance publicly on leaderboards for the whole company or teams could be the reason behind the reason on some social well-being risks. Accessibility refers to accessing all the employee’s earned incentives on their public profiles on the ES, to their colleagues, could be one of the main reasons that could lead to gamification risks. “I feel embarrassed and ashamed of my achieved badges if I am level 1 in something, and I wish to hide them”, leading to embarrassment.

IV. CONCLUSION AND FUTURE WORK

In this work, “Inclusive Gamification” is introduced as a concept that considers contextual elements in gamification. The study presents risk factors and associated risks in the software development industry, taking into account various contextual aspects. This work aims to provide a systematic method for identifying and mitigating gamification risks, and future work will focus on developing an engineering method for evaluating and mitigating these risks in enterprise systems.

REFERENCES

Session SEKE1: Software Engineering and Knowledge Engineering
Will you use software development support using biosignals? A survey from software developers

Ryo Soga*†, Hideyuki Kanuka*, Takatomi Kubo†, Takashi Ishio‡, Kenichi Matsumoto†
* Research & Development Group, Hitachi, Ltd.
{ryo.soga.jj, hideyuki.kanuka.dv}@hitachi.com
† Graduate School of Science and Technology, Nara Institute of Science and Technology
{takatomi-k, matumoto}@is.naist.jp
‡ Department of Media Architecture, Future University Hakodate
ishio@fun.ac.jp

Abstract—Biosignals reflect the mental states of software developers and could improve support technologies for software development activities. Although several technologies for software development support using biosignals (BioSDS) have been proposed, BioSDS has not yet been deployed in actual software development workplaces. As a prerequisite for industrial deployment, BioSDS must be well understood and accepted by software developers. However, the current level of their acceptance has not been comprehensively assessed. In this study, we conducted a survey to clarify the current level of acceptance of BioSDS and potential attributes that influence the level of acceptance. We defined eleven use-cases based on six previous primary studies related to BioSDS, and then asked developers at Hitachi, a Japanese IT company in the FORTUNE 500, about the level of acceptance of each use-case. Our analysis of eighty-six responses revealed that four out of eleven use-cases had some level of acceptance by software developers. In addition, we found four attributes that affect the level of acceptance: subject to be measured, objectives, interventions, and timing. These findings help to identify barriers to the adoption of BioSDS in the workplace.

Index Terms—Biosignal, Acceptance, Software development support

I. INTRODUCTION

Several technologies for software development support using biosignals (BioSDS) have been proposed but have not yet been deployed in workplaces of software development. For example, Züger et al. showed that software developers’ interruptibility can be estimated from biosignals and proposed a use-case (UC) in which software developers can easily maintain a focused state by notifying their interruptibility to teammates [18]. Müller et al. also showed that biosignals can be used to estimate whether a developer’s edits contain defects and proposed a UC that recommends peer review of codes containing defects [13]. These UCs can boost software development by considering the mental state of software developers [4].

As the prerequisites for the introduction of BioSDS to workplaces, software developers need to accept BioSDS. The acceptance to such new technologies has been described following the technology acceptance model (TAM) [2]. Following TAM, the acceptance of BioSDS could be evaluated from the agreement level with the the self-predicted future usage.

In this study, we conducted a survey of software developers at a large Japanese IT company to determine the current level of acceptance of BioSDS and to explore attributes that influence the acceptance. We analyzed BioSDS UCs presented by six previous primary studies and extracted four candidate attributes of UCs affecting the acceptance: (i) subject to be measured, (ii) objective, (iii) intervention and (iv) timing. We defined 11 BioSDS UCs varying attributes (ii)–(iv), and designed questionnaires that asks the acceptance level of each UC. In addition, for (i), two types of questionnaires were created: whether the subject was the respondent or not.

We obtained 86 responses and analyzed them to answer the following two research questions (RQs).

RQ1. Do software developers accept BioSDS?
RQ2. How do attributes of UCs affect the level of acceptance?

These findings contribute to reveal the obstacles to deploy BioSDS in workplaces.

II. RELATED WORK

A. Review of software engineering studies using biosignals

Several review studies were presented previously [6], [10], [15]. Weber et al. identified 89 studies in the field of software engineering that used neural activity, such as brain activity and autonomic nervous systems, and showed that there are 4 types of contributions that can be made by the study of neural activity, [15]: (1) contributions to understanding human factors (2) improvement in software development, (3) investigation of software-understanding methods, and (4) development of software systems that adapt to the user’s mental state. Menzen et al. identified 40 major studies that used biosignals in software engineering, categorized their types and themes, and pointed out the lack of application in real-world development environments [10]. Gonzales et al. selected 33 studies that used
biosignals to estimate cognitive load and investigated the
challenges in realistic scenarios, noting the lack of accuracy
in machine learning, [6]. These studies are similar to ours
in that they compared studies that used biosignals in soft-
ware engineering and attempted to identify challenges in
existing research. However, they did not consider whether
they are accepted by software developers.

B. Evaluating software-development support technologies

Several studies proposed new support technologies and
evaluated usefulness of them [1], [5], [17]. In a study that
did not use biosignals, Züger et al. asked 449 knowledge
workers from 12 countries to use a light to make it known
they did not want to be disturbed [17]. During the first
half of the five-week experiment, they conducted a survey
asking whether the participants wanted to stop using the
light and indicated their willingness to continue after they
were allowed to use the light. They were also surveyed
about whether they wanted to continue using it. There are
two studies that used eye gaze as a biosignal. Glücker et
al. implemented EyeDE, which uses eye trackers to browse
related parts of a program displayed in an integrated de-
velopment environment (IDE) with the gaze. Four people
were interviewed about their qualitative impressions of
EyeDE and found it interesting [5]. Ahrens et al. also
developed a tool to assist novice users in understanding a
program by displaying a heatmap on the IDE that shows
the viewing positions of the eyes viewed by expert users.
Feedback on this tool was diverse, including responses
that it reduced comprehension difficulty and that it was
intrusive, [1]. However, these were only usefulness eval-
uations for individual UCs. Quantitative evaluations when
comparing UCs have not been conducted.

III. Method

The acceptance of BioSDS was surveyed by defining
comprehensive BioSDS UCs and presenting them to soft-
ware developers. Four candidate attributes of BioSDS UCs
were extracted from 6 previous primary studies: (i) sub-
ject to be measured, (ii) objective, (iii) intervention and
(iv) timing. Comprehensive BioSDS UCs were defined as
varying 3 attributes (ii)–(iv). Subsequently, questionnaires
were designed to ask the acceptance of each UC and
and the reasons of selecting the most or the least useful UC.
Besides, for (i), the questionnaires were separately asked
whether the subject was the respondent or not. Eventually,
responses were analyzed to answer 3 RQs.

A. Attributes of BioSDS UCs

1) Select primary BioSDS studies: Six primary BioSDS
studies were selected (Table I) from studies cited by
the most cited review study [15] among previous review
studies of software engineering studies using biosignals [6],

The most cited review study constructed the search
query by combining 22 words related to neural activity
and 13 words related to software engineering, and obtained
89 studies. Then, the review study classified obtained
89 studies into 5 categories: empirical studies (N=47),
empirical (research in progress) (N=24), methodological
studies (N=8), conceptual studies (N=5), and review
studies (N=5). Empirical studies were further categorized
into studies in which biosignals were explanatory variables
(N=19) and those in which biosignals were the target
variable (N=26).

We selected 6 primary studies satisfying two criteria
from 19 studies in which biosignals were explanatory
variables. Firstly, 10 studies were excluded because the
number of citations was less than 10. Secondly, 3 studies
were excluded because they did not explicitly present UCs.
Consequently, 6 studies were remained and their number
of citations were 25–158(Table I). The number of citations
was retrieved from Scopus (https://www.scopus.com) in
May 2022.

2) Extract UCs and their attributes from six primary
BioSDS studies: Fifteen UCs were extracted from six
primary BioSDS studies (Table I). All UCs analyzed
biosignals to detect the stress or the cognitive load of
software developers, and attempted to boost software
development by interventions. From 12 UCs, we extracted
four candidate attributes: (i) subject to be measured, (ii)
objective, (iii) intervention and (iv) timing.

(i) All UCs had two types of subjects: subjects to be
measured and subjects not to be measured and two types
of subjects would have totally different experiences in UCs.
For example, “Assistance in preventing interruption” had
at least two subjects and one subject was measured and
prevented from the other’s interruptions. Therefore, the
acceptance would be varying by whether “subject to be
measured = respondent” or “subject to be measured =
non-respondent”.

(ii) The UCs’ objectives were categorized into following
eight types. “O1. Assessing quality”, which had 2 UCs,
estimated the quality of programs and attempted to iden-
tify codes to be reviewed. “O2. Assessing skills”, which
had 1 UC, estimated the skills of software developers.
“O3. Preventing bugs”, which had 5 UCs, estimated the
quality of programs and attempted to prevent bugs by
double-checking the program. “O4. Preventing interrup-
tions”, which had 2 UCs, estimated the interruptibility
of software developers and attempted to prevent inter-
ruptions from other software developers. “O5. Searching
code”, which had 1 UC, estimated the cognitive load of
software developers and attempted to present programs
related to codes causing the high cognitive load. “O6.
Taking breaks”, which had 2 UCs, estimated the stress of
software developers and attempted to encourage software
developers to take a break. The acceptance would be
varying by whether each objective satisfied needs of each
software developer.

(iii) The UC’s interventions were categorized into 3
types, and each type of interventions had different impact
on software development. Both “I1. Notification (Private)” and “I2. Notification (Team)” intervened software development by notifying estimation results to individuals or teammates, and their impacts would be relatively low because software developers could freely ignore notifications. “I3. Evaluation” intervened software development by evaluating skills or the quality of programs and its impact would be relatively high because additional works would be required when the evaluation results was bad. The acceptance would be varying by whether software developers tolerated those impacts.

(iv) The UC’s timings of intervention were categorized into 2 types: “Real-time” and “Non-real-time”, and two types of timings varied frequencies and adequacy. Interventions in real-time could help software developers immediately when they had troubles, but too much interventions could ironically cause additional stress or interrupt self-help efforts. Therefore, the acceptance would be varying depending on the intervention timing.

B. Comprehensive BioSDS UCs

Based on extracted 12 BioSDS UCs and 4 attributes, 11 UCs were defined (Table II). Six UCs were defined to consider the most common objective “O3. Preventing bugs” (Table II UC03, UC04, UC06, UC07, UC10, UC11) and 5 UCs were defined to consider other 5 different objectives (Table II UC01, UC02, UC05, UC08, UC09).

Six UCs related to “O3. Preventing bugs” were defined as varying (iii) interventions and (iv) timing. Four UCs were defined by combining two types of interventions: “I1. Notification (Private)” and “I2. Notification (Team)”, and two types of intervention timing: “Real-time” and “Non-real-time” (Table II UC03, UC04, UC06, UC07). Besides, to further investigate how interventions affect the acceptance, we added the intervention category “I4. Edit” and defined 2 original UC scenarios (Table II UC10, UC11).

C. Questionnaires

We created two descriptions for each UC with different subjects to be measured: respondent and non-respondent. For the UCs whose subject is respondent, we use the descriptions in Table II. For the UCs whose subject is non-respondent, we replaced “(you)” with “(someone other than you)” in the descriptions. Each description is provided with a Likert question that asks the acceptance level of the description. We asked the same questions separately for those two groups of descriptions to understand the impact of the subject to be measured.

A Likert question was designed to evaluate “self-predicted future usage” as the acceptance. The actual question was “If the technology were available, would you use it in your future tasks?” and had 5 options: (1) “strongly disagree”, (2) “disagree”, (3) “neutral”, (4) “agree”, (5) “strongly agree”. Likert responses higher than 3 suggest acceptance and that lower than 3 suggest rejection.

Questionnaires consisted of three pages. The first page provided the summary and prerequisites of survey. As the prerequisites for survey, “Do not care the feasibility” was written because software for realizing any of defined UCs is not yet available although software for collecting biosignals is available [11], [14]. The second page asked the Likert questions for descriptions whose subject to be measured is respondent. The third page asked the same questions for descriptions whose subject to be measured is non-respondent. All sentences in questionnaires were written in Japanese.

D. Analysis

1) RQ1. Do software developers accept BioSDS?: If software developers accept a UC of BioSDS, their Likert responses for a UC scenarios should be greater than 3 (neutral). We conducted Wilcoxon signed rank test [16] to analyze whether the median of Likert responses was significantly greater than 3 or not.
TABLE II
UCs created in this study

<table>
<thead>
<tr>
<th>#</th>
<th>Description of UCs</th>
<th>Reference</th>
<th>Objective</th>
<th>Intervention</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC01</td>
<td>Prompt the developer (you) to take a break when the developer (you) feels stressed.</td>
<td>[12]</td>
<td>O6. Taking breaks</td>
<td>I1. Notification (Private)</td>
<td>Real-time</td>
</tr>
<tr>
<td>UC02</td>
<td>Present a candidate related code to the developer (you) only when the developer (you) feels stressed because it is difficult to find a related code.</td>
<td>[12]</td>
<td>O5. Searching code</td>
<td>I1. Notification (Private)</td>
<td>Real-time</td>
</tr>
<tr>
<td>UC03</td>
<td>While the developer (you) is viewing or modifying a code, identify the code that caused the stress in real-time and encourage the developer (you) to review the modification or consult with others.</td>
<td>[3], [7], [9], [13]</td>
<td>O3. Preventing bugs</td>
<td>I1. Notification (Private)</td>
<td>Real-time</td>
</tr>
<tr>
<td>UC04</td>
<td>Prompt the developer (you) to review the changes or consult with others before reflecting the changes in the production code after the developer (you) has completed modifying the code.</td>
<td>[3], [7], [9], [13]</td>
<td>O3. Preventing bugs</td>
<td>I1. Notification (Private)</td>
<td>Non-real-time</td>
</tr>
<tr>
<td>UC05</td>
<td>Let teammates know in real-time when the developer’s (your) concentration level is high to prevent interruptions from those around the developer (you).</td>
<td>[12], [18]</td>
<td>O4. Preventing interruptions</td>
<td>I2. Notification (Team)</td>
<td>Real-time</td>
</tr>
<tr>
<td>UC06</td>
<td>While the developer (you) is modifying the code, identify the code that caused the stress in real-time and inform co-editors or teammates of the high stress of the developer (you).</td>
<td>[12], [13]</td>
<td>O3. Preventing bugs</td>
<td>I2. Notification (Team)</td>
<td>Real-time</td>
</tr>
<tr>
<td>UC07</td>
<td>Prompt the reviewer to conduct a focused code review of the code that caused the developer (you) stress after the developer (you) has completed the task but before it is reflected in the production code.</td>
<td>[12], [13]</td>
<td>O3. Preventing bugs</td>
<td>I2. Notification (Team)</td>
<td>Non-real-time</td>
</tr>
<tr>
<td>UC08</td>
<td>Estimate the quality of the refactoring based on the stress while the developer (you) is viewing the code to evaluate how the readability is improved from the developer’s (your) perspective.</td>
<td>[7], [12]</td>
<td>O1. Assessing quality</td>
<td>I3. Assessment</td>
<td>–</td>
</tr>
<tr>
<td>UC09</td>
<td>Estimate technical skills on the basis of the stress while the developer (you) is viewing and modifying the code.</td>
<td>[9]</td>
<td>O2. Assessing skills</td>
<td>I3. Assessment</td>
<td>–</td>
</tr>
<tr>
<td>UC10</td>
<td>Edit code conventions to prevent program patterns that caused the developer (you) stress by identifying the code that caused the stress in real-time while the developer (you) is viewing the code.</td>
<td>–</td>
<td>O3. Preventing bugs</td>
<td>I4. Edit</td>
<td>Real-time</td>
</tr>
<tr>
<td>UC11</td>
<td>Submit issues for refactoring the code that caused the developer (you) stress by identifying that code in real-time while the developer (you) is viewing the code.</td>
<td>–</td>
<td>O3. Preventing bugs</td>
<td>I4. Edit</td>
<td>Real-Time</td>
</tr>
</tbody>
</table>

2) RQ2. How do attributes of UCs affect the acceptance?: The difference in Likert responses was analyzed depending on the four attributes of UCs: (i) subject to be measured, (ii) objectives, (iii) interventions, and (iv) timing. For (i) the subject to be measured, the responses were divided into two groups, and Wilcoxon signed rank test was conducted as a test for two related paired samples. For (ii) objectives and (iii) interventions, the response values were divided into multiple groups, and the Kruskal-Wallis test [8] was conducted as a multiple comparison test. When the test result was significant, Wilcoxon rank-sum test was conducted for every two groups. For (iv) timing, to conduct a test for two related paired samples, we limited the objective to “O3. Preventing bugs” and the intervention of “Notification (I1, I2)”, and Wilcoxon signed rank test was conducted.

E. Distribution

The survey was distributed to employees at Hitachi, Ltd., a large Japanese IT company in FORTUNE 500; the company has more than 3,000 employees and has been doing business for more than 100 years in the electric device field. The survey was distributed to about 100 software developers in the financial industry and about 60 employees in the research department.

IV. Results

We obtained 86 responses, of which 52 were from the development sector and 34 from the R&D sector. The average response time was 38 minutes. The respondents’ years of work experience were widely distributed from 0 to 35 years, with a particularly large number of respondents around 0 and 20 years (Fig. 1A). This indicates that this survey could reflect the opinions from both novice and experienced software developers.
A. RQ1. Do software developers accept BioSDS?

We verified whether the median Likert response of each of the 11 UCs was greater than 3 when the subject to be measured was either a respondent or non-respondent (Fig. 1B). As the result, the median was significantly greater than 3 for 4 use-cases (UC01, UC02, UC05, and UC07), and for 1 use-case (UC11) the median was significantly less than 3. This result indicates that developers accept some BioSDS UCs.

B. RQ2. How do attributes of UCs affect the level of acceptance?

For each of the 4 attributes of UC defined in this study, we grouped Likert responses by attribute values and compared median values of each group (Fig. 1C).

1) Subject to be measured: The median Likert response was significantly higher (Fig. 1C(i), \( p = 5.34 \times 10^{-5} < 0.0001 \)) when the subject to be measured was non-respondent than when the subject was respondent. This indicates that the acceptance is higher when the software developer is not measured.

2) Timing: The median Likert response was significantly higher (Fig. 1C(ii), \( p = 0.00132 < 0.05 \)) when the intervention timing was non-real-time than when the intervention timing was real-time. This indicates that developers could be more accepting of BioSDS with “Notification” interventions (I1, I2) and “O3. Preventing bugs” objective.

3) Objective: The median Likert response showed a significant difference among 6 types of objectives (Fig. 1C(iii), \( p = 2.83 \times 10^{-5} < 0.0001 \)). Among fifteen pairs of two of the six objectives, 3 were significantly different after Bonferroni correction: \( O1 < O5 (p = 0.00571 < 0.05), O2 < O5 (p = 0.00622 < 0.05), \) and \( O3 < O5 (p = 0.00142 < 0.05) \). This indicates that “O5. Searching code” is the most acceptable, and “O4. Preventing interruptions” and “O6. Taking breaks” breaks” are moderately acceptable.

4) Intervention: The median Likert response showed a significant difference among 4 types of interventions (Fig. 1C(iv), \( p = 1.19 \times 10^{-12} < 0.0001 \)) as well as the objectives. Among six combinations of two of the four interventions, four combinations were significantly different after Bonferroni correction: \( I1 > I3 (p = 0.00148 < 0.05), I1 > I4 (p = 3.11 \times 10^{-10} < 0.0001), I2 > I3 (p = 0.00627 < 0.05), \) and \( I2 > I4 (p = 1.26 \times 10^{-8} < 0.05) \).
This indicates that the two “Notification” interventions (I1, I2) are more acceptable than the other interventions (I3, I4).

V. DISCUSSION

We examined the level of acceptance for each of the UCs based on previous BioSDS studies, and revealed the current level of acceptance (RQ1) and how four attributes affect the level of acceptance (RQ2).

First, developers accepted some BioSDS UCs to a certain extent (Fig. 1B). This result suggests that at least some BioSDS UCs are ready to be deployed in the workplace from an acceptance perspective. Currently, if developers want to start using BioSDS, they cannot try out BioSDS UCs because it is difficult to prepare machine learning models trained by BioSDS studies. In addition, some measurement devices, such as eye trackers, are too expensive for all software developers to install. In the future, it is expected that at least some of the use-cases can be deployed in the workplace by publishing research-trained models or by providing inexpensive devices.

Second, all 4 of the attributes defined in this study affect the level of acceptance (Fig. 1C). For each of these attributes, respondents prefer values where the impact is relatively small. For example, acceptance is higher when the software developer is not measured (Fig. 1C(i)). In the future, we should clarify the reason by conducting a qualitative analysis.

VI. CONCLUSION

The acceptance of software development support technologies using biosignals (BioSDS) was investigated by presenting eleven defined use-cases to software developers in a large Japanese IT company. As a result of the analysis of 86 responses, 2 findings were obtained.

1) Four out of 11 BioSDS use-cases (UC01, UC02, UC05, UC07) were accepted by the software developers.

2) The level of acceptance of BioSDS use-cases varied depending on four attributes: the subject to be measured, the objectives, the interventions, and the timing. For all these attributes, software developers preferred low-impact values for them.

These results suggest that at least some use-cases are ready to be deployed in the workplace from an acceptance perspective, and deployment could be advanced by publishing assets created in BioSDS studies. On the other hand, the acceptance of BioSDS use-cases with high impact was still low, and we should clarify the reason by conducting qualitative analysis in the future.

As a limitation of this study, it is not clear how acceptance varies depending on the characteristics of software developers, as the respondents were a heterogeneous group. In the future, we will use a mathematical method for clustering heterogeneous groups to reveal which type of software developers tend to be more accepting of the use of BioSDS.

ACKNOWLEDGMENTS

This work is supported by JSPS KAKENHI Grant Numbers JP20H05706.

REFERENCES

Assuring Domain Software Quality through Workflow Testing and Specification

Melody L. Hammel, Lan Lin
Department of Computer Science, Ball State University, Muncie, IN 47306, USA
{*mlhammel, llin4*}@bsu.edu

Abstract

In this paper, we report our experience testing the GenericModelAgent toolkit, a critical piece of the software developed for the NSF CyberWater project [1], to ensure that the workflows work properly without error when deployed to the user community. The many challenges we faced include a lack of developer-oriented specifications, a complex framework, complicated and domain-specific functionality, frequently changing requirements, domain developers’ unfamiliarity with TDD best practices, and poor model-view separation in a legacy software used to construct visual workflows. We present a generalized testing strategy for approaching different modules in this toolkit, in an effort to understand every input, its relation to the main functionality of the module, and every output. We then built unit and integration test cases from bottom up to enable systematic and thorough exercises of all usage scenarios identified of every module as well as their integration. Our testing discovered problems in the toolkit with suggested fixes that no one was aware of before. It also led to the creation of rewritten specifications for each module, as a by-product, that are precise and developer- and tester- (rather than user-) oriented. Our specifications, test plan, test cases and results became reusable assets for future development, testing and maintenance of the toolkit.

1. Introduction

CyberWater is a project in development funded by the National Science Foundation (NSF) and designed to reduce user time and effort required for hydrologic modeling studies. According to CUAHSI [1], CyberWater allows hydrologists with little to no coding knowledge to integrate their models into the system and begin running simulations, allowing fundamental discoveries to be made more frequently, more easily and with less effort than before. Data flow is automated and easy access to High-Performance Computing (HPC) is provided, allowing more efficient execution and study of hydrologic data models and simulations, significantly increasing the accessibility of open-data workflows and lowering the learning curve and barrier to entry [1]. The development team consists of developers from a wide array of backgrounds, but not all of them are familiar with software engineering best practices.

One crucial component of their delivered software is the GenericModelAgent toolkit [4], which is a collection of modules within the CyberWater framework on top of VisTrails [2]. VisTrails is a GUI-based software for building scientific workflows using blocks called modules, which connect to each other to pass outputs to another module’s inputs. These are called output ports and input ports respectively. It is typically used for data visualization and running simulations. An example workflow is shown in Figure 1.

![Figure 1. An example workflow in VisTrails using the GenericModelAgent Toolkit](image)

DOI reference number: 10.18293/SEKE2023-063
combined workflow. Thus, they have functionality within each that relies on the proper functionality of the previous module in the chain, requiring a top-down approach to understand them – knowing what the previous module in the workflow does is almost always integral to understanding the functionality of the next. Though we faced this and plenty of other challenges in attempting to test these modules to ensure the quality of the final product, we were able to overcome said challenges and produce work products that became reusable testing assets, such as reworked specifications, fixed bugs, thorough unit and integration tests and test documentation, as well as insight as to how this could have been done better and made easier in the future.

2. Challenges of Testing

The GenericModelAgent toolkit is a critical component of the CyberWater framework software, providing major functionality of the advocated open-data open-model framework, and thus needs to be thoroughly tested and assured to be of good quality. When we were asked to test these modules, there were a few heuristic test cases written by the developer, but they are not documented, saved in the repository, automated, nor part of the CI/CD process.

However, with GenericModelAgent already deployed in users’ hands and limited testing to be found, test cases had to be designed quickly and efficiently to reveal any hidden bugs in the software. Our job was to thoroughly test this toolkit, writing and automating both unit and integration tests in order to ensure the proper functionality of all the modules in this toolkit and integrate CI/CD into the workflow. In doing so, we have learned much about the design of this toolkit and have helped to improve it, both in code quality and in catching hidden bugs.

As the toolkit is already in use, any change made to the code will have an almost immediate effect. Unit testing is integral, in this scenario, to ensure that no broken or less-than-quality code makes its way into a part of the toolkit that end-users will have. Even beyond that fact, however, the toolkit is built as an extension of VisTrails [2], legacy software which is written in an older, deprecated version of Python (version 2.7), and lacks model-view separation in some areas, making testing without using the GUI somewhat difficult [5].

There were a few challenges that we faced at the beginning of testing:

1. Lack of developer/tester-oriented specifications for the modules and their functionality

With a lack of both developer-oriented specifications and knowledge in the field of water science, for which this software is being developed, much of the testing process for each module involved attempting to understand what it was we were testing for. We needed to delve deep into the outputs of the module as compared to the inputs. This led us to find that the data and inputs of the modules were much more complicated than we had previously thought.

2. A large, complex, inter-woven framework upon which these modules are built

This toolkit was designed heavily around the MSM package [6], legacy software and a predecessor upon which the CyberWater framework was built and which has a winding, interconnected set of modules and functionalities upon which GenericModelAgent was based. That means fully understanding GenericModelAgent requires also fully understanding MSM – though this was not within the scope of our testing.

3. Connections between modules and the complication of their functionality

Similar to MSM, this toolkit involves a set of modules which are closely related to each other, given that their purpose is to be used in tandem in a workflow in VisTrails. Thus, they all assume functionality of each other, where a later module in the workflow will presume that the previous module has done something special – editing one module’s functionality could potentially require changing all the modules’ functionalities. The functionality in itself is complicated as well, given that it spans across modules – the high level idea is hidden unless one understands the domain specificity of the intended use of this toolkit.

4. Extremely domain-specific functionality

The functionality of these modules is very domain-specific. Most of the functionality that does not revolve around setting up a directory structure involves processing hydrological data, with all the data being passed around in structures referred to as datasets. Given that data processing is unavoidable and the specifications we received didn’t go deep into the data, understanding the structure of it and what each point of it meant was integral to testing these modules, as we needed to create consistent test data for the modules to work with.

5. Frequently changing functionality in the midst of testing

Since this toolkit is both actively in development and deployed and in use by end-users, we also found that the functionality was changing, even if slightly, throughout the course of our testing of these modules – even in subtle ways, such as changing the way the
output data is formatted, required us to rework our test cases around the new intended functionality. With a lack of specification as to what the output of these modules was really supposed to look like, the new, updated functionality was stated as a fact and our test cases and design had to be adapted to the new version of the modules.

6. Lack of development using TDD, and what testing that did exist was heuristic and not thorough

The things we know as software engineers to be best practices such as Test-Driven Development (TDD) [3] and proper Git flow would have made the whole process smoother – tests would have been developed for the modules as the modules themselves were developed, essentially creating a specification for themselves. Testing that did exist was primarily heuristic. The developers were unfamiliar with unit testing, and often misunderstood what it was that we needed to perform black-box unit testing. Similarly, proper Git flow was rarely used, and the rapidly-changing functionality was made more difficult to keep track of.

7. Lack of model-view separation in VisTrails’s module design

VisTrails is legacy software. Its design of the modules lacks proper model-view separation, making testing in the back-end difficult – similarly, the GenericModelAgent toolkit is primarily visual software, designed around processing data to be visualized by the MSM package. Thus, we were tasked with automating unit testing in the back-end for software designed to be a visual workflow – our test design had to be generalized to extend to any type of workflow, allowing us to ensure the quality of both our methods and of the software under test.

Despite all the challenges, we accomplished what we set out to do. We developed a systematic approach to testing every individual module in this toolkit, documented our solution and produced meaningful results to aid in the development as well as to improve the quality moving forward, which we elaborate in the next section.

3. Our Solution to the Testing Problem

The GenericModelAgent toolkit, as shown in Figure 2, consists of five modules:

- MainGenerator
- AreaWiseParamGenerator
- InitialStateFileGenerator
- ForcingDataFileGenerator
- RunModuleAgent

The workflow is designed around the concept of running a user’s “model.” Typically, this represents a compiled, executable file either downloaded or written by the user, which is designed to take in some input data and output new data into files with identical structure to the input files. The goal of GenericModelAgent is to simplify the process of setting up the environment needed to run the model, allowing the user to get their model up and running quickly and without having to write any glue code. The model is easily integrated using the MSM package to visually represent the data output by the model’s simulation [4].

When beginning testing of this toolkit, we had already tested the first module in the workflow, MainGenerator, so we already had a decent idea of how testing of the rest of the workflow would go. MainGenerator, however, was a comparatively simple module – we consider it in greater detail in [5]. “Generator” modules typically have simpler but limited functionality, revolving around setting up the directory structure. A “Generator” module’s output WD_Path will be used as the input to all other modules’ inputs of the same name, providing the working directory of all the functionality in the workflow.

One by-product of our testing was the production of precise specifications, both in the form of an overview using one or more visual diagrams displaying the overall functionality and inputs/outputs of the module as well as detailed written developer documentation, as shown in Figure 3. In essence, it serves as a guide for future specifications to be written and a good reference for what to test when writing unit tests for these modules. These specifications were developer-focused and low-level, to enable black-box testing without exposing any implementation details.

We developed a strategy that we applied to all the modules in the toolkit in order to understand the environment in which they work and their functionality. We first began by carefully considering every input and output that the module can receive and generate – this extends beyond just input and output ports and to other things that the module...
AreaWiseParamGenerator: An Updated Spec

* AreaWiseParamGenerator, from an end-user standpoint, organizes files to be used in the rest of the workflow. It does this by placing a number of input files (up to 36) into a specific folder within the toolKit’s working directory using the compute method, just as all other VisTrails modules

AWPG: Inputs

* WD_Path (VT String): The working directory in which to create the folder, provided by MainGenerator’s output port of the same name. If no input is provided or the path provided can’t be accessed, an exception should be raised.
* Parameter_folder_Name (VT String): Name of the folder to create inside WD_Path, into which the file will be copied. No input is provided, the default value is “Parameters”. If the folder already exists, use that one, but throw an XSIError if we can’t access it.
* File_In_XX (VT String or VT File): File path to a file to be copied into the folder created by Parameter_folder_Name. _XX replaced by any of _00_—_15 for the names of all 15 ports

AWPG: Output

* Ready (bool): If the module completes all its operations successfully, this value reads “true” and otherwise should be “false”. Passed to RunModuleAgent in the full workflow, and interrupts the workflow if this isn’t “true”.

Figure 3. A snippet from our rewritten AreaWiseParamGenerator specification

can receive: directory structure, files in a location it expects them to be, a previous module’s execution as inputs; and resulting directory structure, generated files, copied files as outputs. We then examined how each one of these inputs and outputs relates to the overall behavior (both intended and implemented) of the module, asking deeper questions such as how does this input affect the output? What is done with any given input that allows this output and behavior to be observed? How changes in inputs bring about different outputs?

3.1. AreaWiseParamGenerator and InitialStateFileGenerator

These are two more “Generator” modules in the toolkit and, as “Generator” modules, both are similar in function to MainGenerator – designed around setting up the environment for the user model to run. We designed tests for success and failure of each individual function, with various types of edge cases, such as file system permissions and an improper directory structure. Keeping consistent test data, such as making sure file names and folder names are always the same, helped drastically in making the tests cohesive. The “Generator” modules (InitialStateFileGenerator, as an example, shown in Figure 4) only have one output port, designed to connect them together in a VisTrails workflow. The outputs of the modules boil down to two main things: a directory structure and raised errors.

Since VisTrails modules are based around their compute() methods, our testing strategy was to define constant test data for things not being tested, call the compute() method and assert that the directory structure looked how we expected it to after the compute() method finished. With each test, we varied only the input to the one port or aspect of behavior, shown in Table 1. We considered every scenario that could go wrong and grouped tests based on input and output ports.

3.2. ForcingDataFileGenerator and RunModuleAgent

These two are a spike in complication, both in function and in testing: they start to integrate with the MSM package, involving small amounts of data processing. Thus, we needed to understand the data, the outputs and what inputs should cause what outputs. This was increasingly difficult with the lack of developer documentation and specification. This pushed most of our focus onto understanding the inputs and outputs.

In order to run automated unit tests in the back-end for workflows built in VisTrails, we utilized object-method replacement, (used in the testing of MainGenerator [5]). To apply it to testing the modules here, we had to revamp the get_input method to handle “compound input ports.”

The MSM package uses a structure called DaoDataSet. These are objects in Python which, at their core, represent JSON files with additional methods to operate on the data contained within. These JSON files have a deeply nested structure, shown in Figure 5, with many keys and values required for the DaoDataSet constructor to function. In order to test these modules which use these datasets, we needed to know the structure of these datasets, to create test data that will result in predictable outputs in testing. It took much experimentation and effort.
Table 1. Test design for \texttt{AreaWiseParamGenerator} and \texttt{InitialStateFileGenerator}. Scenarios marked with asterisk (*) are merged into other test cases in practice.

<table>
<thead>
<tr>
<th>AreaWiseParamGenerator</th>
<th>InitialStateFileGenerator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Param.Folder.Name</strong></td>
<td><strong>Init.State.Folder.Name</strong></td>
</tr>
<tr>
<td>Folder exists with write permission*</td>
<td>Folder exists with write permission*</td>
</tr>
<tr>
<td>No name provided</td>
<td>No name provided</td>
</tr>
<tr>
<td>Folder doesn’t exist</td>
<td>Folder exists w/o write permission</td>
</tr>
<tr>
<td>Folder exists w/o write permission</td>
<td>Previous folder contents removed</td>
</tr>
<tr>
<td><strong>WD.Path</strong></td>
<td><strong>WD.Path</strong></td>
</tr>
<tr>
<td>No path provided</td>
<td>No path provided</td>
</tr>
<tr>
<td>Path doesn’t exist</td>
<td>Path doesn’t exist</td>
</tr>
<tr>
<td>Path exists w/o write permission*</td>
<td>Path exists w/o write permission</td>
</tr>
<tr>
<td><strong>File.In.XX</strong></td>
<td><strong>File.In.X</strong></td>
</tr>
<tr>
<td>No file provided</td>
<td>No file provided</td>
</tr>
<tr>
<td>File doesn’t exist</td>
<td>File doesn’t exist</td>
</tr>
<tr>
<td>File exists w/o read permission</td>
<td>File exists w/o read permission</td>
</tr>
<tr>
<td>File exists with read permission</td>
<td>File exists with read permission</td>
</tr>
<tr>
<td>Ready signal (Out)</td>
<td>Ready signal (Out)</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

as little was well-defined.

In most instances, test case design for \texttt{ForcingDataFileGenerator} had to revolve around crafting various test datasets as inputs with varying structures of data that would cause different testable behavior and asserting that it behaved in an expected way. We considered every possible varying output of the module and chose specific structures out of the wide variety of datasets to ensure correct output was produced for the given scenario.

\texttt{RunModuleAgent}’s functionality, shown in Figure 6, is based around the concept of a “model.” This is assumed to be a program which can be run from the command-line with arguments, which generates files of a specific name and format to be read by \texttt{RunModuleAgent} and placed into a \textit{dataset}. We wrote a script to run the module in our tests that had predictable outputs and wasn’t computationally intensive.

### 3.3. The Test Model

The test model developed for testing these modules is written in C, to be compiled at run-time by a \texttt{system()} call in the Python script that runs the tests and to be called by \texttt{RunModuleAgent}. It is programmed with multiple different outputs that it can generate to test the different responses of \texttt{RunModuleAgent} and the workflow, which can be specified by the argument with which the model is run. For example, if the model is run with \texttt{multiple_columns} as an argument, the file it generates will have multiple columns — however, if its argument is \texttt{alternate_separator}, the file it generates will be delimited by commas instead of tabs. The model generates an output file in different formats depending on the argument it is run with. This model, since compiled at runtime, is also customizable, and any of its parts can be switched in and out or ignored with ease, due to the use of arguments – simply add an extra else case with a string comparison to the code. A snippet of the model’s code is shown in Figure 7.

### 3.4. Integration

After unit testing all of the modules thoroughly and individually, integration testing was an indispensable step in ensuring the functionality and code quality of this toolkit. Previously, we had only tested these modules in isolation, making sure they produce the outputs we expect given specific inputs, but integration testing allows us to test them in a similar environment to the one they will be used in: providing each other inputs through one another’s outputs.

Using Python’s \texttt{Unittest} library, we tested an integration of these modules, using one module’s output as the input to another module, simulating how it would be done with connectors in the \textit{VisTrails} GUI, shown in Figure 8.

We used the test model from earlier, extending it to account for every module’s outputs. This then generates an output file with specific information in it depending on which modules failed, if any, and this file is read by the Python tests, interpreted and the user is notified of any failures and the exact module that was the culprit. For example, consider \texttt{ForcingDataFileGenerator} generates a file with incorrect content compared to what we would expect based on the dataset provided. The test model will read the file, compare it against the expected output and generate its own file, appending "fdfg" to the output. The code is shown in Figure 7.

The Python test code will then, upon raising an \texttt{AssertionError} that the output is not what was expected, read said file, as it is now aware an error has taken place and parse the content based on the included text in the file, see that "fdfg" is included and raise the caught \texttt{AssertionError} again with an er-
Figure 5. A snippet of the nested structure of the DaoDataSet JSON

error message explaining to the end-user what went wrong: ForcingDataFileGenerator improperly generated its file. The code for this is shown in Figure 9.

The integration testing ensures that all of the modules can “talk” to each other in ways that each module is expecting and successfully execute a workflow all working in tandem. This provides us an extra layer of confidence of the modules in their expected environment of use.

4. Testing Results

In this section, we report the major results we obtained through testing.

4.1. Test Cases and Discovered Errors

Table 2 shows the number of test cases we wrote and ran for each module in the toolkit, how many of them were successful and how many failed. These test cases allowed us to discover certain bugs that were present with all the modules. The developers were not aware of these bugs without systematic unit and integration testing described here.

4.2. Results Format

In this section, we report the major results we obtained through testing.

Table 2 shows the number of test cases we wrote and ran for each module in the toolkit, how many of them were successful and how many failed. These test cases allowed us to discover certain bugs that were present with all the modules. The developers were not aware of these bugs without systematic unit and integration testing described here.

The most common errors that we found in the modules were ones that resulted from edge cases of the module usage, shown in Table 3. These primarily resulted from instances where a module would fail and the rest of the work-
flow would continue, regardless, until the last module informed the user of the problem because the model it was trying to run would fail. Our test cases revealed flaws in error handling for nearly every module and, with code fixes, informed the user in a helpful way.

Our test cases allowed meaningful improvements to be made to the modules that will provide a better user experience to end-users of the software. Improved error handling will better inform the user about errors encountered in the modules (rather than cryptic Python errors). Our improved error handling in these modules allows them to better achieve their goal of simplifying the integration of a model into the CyberWater framework, thus improving the quality and usability of the final product.

### 4.2. Specifications as By-Products

A substantial, though often ignored, benefit of considering unit testing during the development of a project is the creation of good developer specifications. Specifications are an artifact that continues to be useful and evolves from iteration to iteration of the project. Good specifications can clear up, early in development, what features are to be in the software, and from a black-box tester’s point of view, what the inputs and outputs are, how it will likely be used, all of its parameters, et cetera. Once a unit test written based off of these specifications fails, the developer immediately knows what part of the system failed, what part of the specification it violates, and what the output should look like, which can potentially kick-start bug fixing as less time is spent searching for what is causing the unintended behavior. For more complex aspects of these modules, the specifications included usage examples and, as a part of the specifications for each module, we also produced a diagram of its input ports and output ports with brief descriptions of their functionalities, as shown in Figures 4, 6, and 10. We developed these specifications to be, in essence, good examples of what would have been ideal to receive at the beginning of the project. They are by-products as well as reusable testing assets for future evolving iterations.

### 4.3. Improved Error Handling

The primary problems we found with the modules that we were able to discover and fix were lackluster or non-existent error handling. In most instances, the modules were programmed with many assumptions as to the users’ knowledge of the toolkit at play. It wasn’t considered that a user could provide inputs in a different way; the system was unequipped to handle inputs of various structures.

Our testing was able to catch a problem with the “Ready” signal (an output) of every module, which tells the next module in the chain whether its function completed successfully. This was implemented as a way to connect the modules together in a VisTrails workflow, to ensure that the functionality happened in order. In some instances, the order of the code was written such that the exception would be raised before the signal could be set, but in some other instances, the signal was never set at all, and the signal could never be set to False.
Table 3. Software bugs revealed by testing

<table>
<thead>
<tr>
<th>Module</th>
<th>Failed Test Case</th>
<th>Revealed Bug</th>
</tr>
</thead>
<tbody>
<tr>
<td>AreaWiseParamGenerator</td>
<td>Ready signal outputs False when error occurs</td>
<td>Signal could never be set to False</td>
</tr>
<tr>
<td></td>
<td>Non-existent WD, Path input</td>
<td>Folder was created instead of exception raised</td>
</tr>
<tr>
<td>InitialStateFileGenerator</td>
<td>Ready signal outputs False when error occurs</td>
<td>Signal could never be set to False</td>
</tr>
<tr>
<td></td>
<td>Non-existent WD, Path input</td>
<td>Folder was created instead of exception raised</td>
</tr>
<tr>
<td></td>
<td>Non-existent file input</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No input files provided</td>
<td>User not informed of error</td>
</tr>
<tr>
<td>ForcingDataFileGenerator</td>
<td>Ready signal outputs False when error occurs</td>
<td>Signal could never be set to False</td>
</tr>
<tr>
<td></td>
<td>Non-existent WD, Path input</td>
<td>Folder was created instead of exception raised</td>
</tr>
<tr>
<td></td>
<td>Mask_File not formatted correctly</td>
<td>Exception not raised</td>
</tr>
<tr>
<td></td>
<td>Input non-formatted date string</td>
<td>Exception not raised</td>
</tr>
<tr>
<td></td>
<td>Non-default separator input</td>
<td>Separator remained default value</td>
</tr>
<tr>
<td>RunModuleAgent</td>
<td>Ready signal with False input stops execution</td>
<td>Input was not properly validated</td>
</tr>
<tr>
<td></td>
<td>Non-existent WD, Path input</td>
<td>Folder was created instead of exception raised</td>
</tr>
</tbody>
</table>

to ForcingDataFileGenerator, where the dataset structure was very specific – despite this, the error handling for parsing the datasets was unhelpful, raising cryptic JSON errors about extracting keys that didn’t exist. This was caught by our testing and improvements were made to the error handling to better inform the user of malformed datasets.

Lastly, when RunModuleAgent was provided a non-tab-delimited CSV file, it would fail to correctly parse columns, as it assumed the separator would always be a tab character, despite the input port designed to change the separator within the input file. The issue with the “Ready” signal also extended into this module – as it was designed to be the last module in the workflow, it takes in inputs corresponding to the signals of all the previous modules.

The test cases that we produced as a result of our work on this toolkit and the errors that they allowed us to detect in the modules’ functionality will continue to provide value to the development of the GenericModelAgent toolkit moving forward. Our testing work has undoubtedly improved the quality of the toolkit in field use. Our test design strategy can be applied to testing other toolkits of comparable complex design and prove useful to scientists writing domain software.

5. Conclusion

Testing, without doubt, remains one of the most important and effective means to assure the quality of software, although in development for domain science the benefit of testing is easily overlooked, especially given the lack of expertise and experience. We report here our experience testing the GenericModelAgent toolkit, a critical component of the CyberWater framework, from scratch, applying unit and integration testing and back-end model-view separation in a CI/CD workflow. Our strategy in approaching workflow testing in the back-end, thorough test case design from functional requirements that addresses all the usage scenarios with edge cases, has led to a comprehensive testing documentation for this toolkit as well as bug fixes that no one on the development team would have discovered otherwise. Beyond improving the quality of the final product to be released, we also produced useful specifications of the modules under test that will benefit future iterations of development. The importance of such developer- or test-oriented specifications cannot be overstated.

6. Acknowledgments

This work was generously funded by the National Science Foundation (NSF) under Grants 1835602 and 2209834. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF. Special thanks to Ranran Chen and Dr. Yao Liang at IUPUI for their helpful assistance in understanding the toolkit and the modules within it.

References

Abstract—Graph neural network is an effective deep learning framework for learning graph data. Existing research has introduced different variants of graph neural networks into the field of software defects and has achieved promising results. However, the graph neural network model based on the previous research is essentially transductive, is applied to a single fixed graph, and often ignores the direction and weight of the edges when modeling the network. In practice, software systems are dynamically evolving. Furthermore, in software network modeling, the direction and weight of edges are factors that are worth considering. Based on an inductive graph neural network, we proposed an improved defect prediction method named GSAGE2defect. We first constructed the class dependency network of the program and then used node2vec for embedding learning to automatically obtain the structural features of the network. Then we combined the learned structural features with traditional software code features to initialize the properties of nodes in the class dependency network. Next, we fed the dependency network to GraphSAGE for a deeper class representation. Finally, we evaluated the proposed method based on eight open-source programs and demonstrated that GSAGE2defect achieves an average improvement of 2.09%-26.69% over state-of-the-art methods in terms of F-measure.

Keywords-component; Software defect prediction; Graph neural network; Network embedding; Class dependency network

I. INTRODUCTION

Graph-structured data are ubiquitous nowadays in many domains such as social networks, cybersecurity, and bio- and chemo-informatics [1]. Through the learning of graph data, many tasks in these domains have been successfully solved, such as recommendations in social networks, and drug discovery in biological information networks. In the field of software engineering, software systems can also be abstracted as a Class Dependency Graph (CDG) with classes as nodes and class dependencies as edges. Some researchers have confirmed that learning the CDG structural information by using complex network theory can effectively improve software defect prediction [2-4]. In recent years, researchers have begun to employ deep learning techniques to automatically encode the dependency graph structure into low-dimensional vector spaces to improve downstream software tasks [2].

Many high-performance graph neural networks have been proposed, in which node adjacency information and node attributes are combined to capture structural information well (e.g., [3,5-7]). It is not difficult to find that the Graph Neural Network (GNN) models, based on which previous studies have been carried out, are inherently transductive and have only been applied in settings with a single fixed graph. However, graph structure data in real scenarios are often dynamic. As we know, a successful software system usually undergoes multiple consecutive versions during its life cycle. In other words, the structure of a software system continues to evolve with requirements and other factors. For instance, the fixing of bugs and the updating of functions may result in the addition or deletion of classes. The transductive-based GNNs primarily generate node embeddings on fixed graphs, and for new nodes, they usually require relearning the new graph, even if it is just a small update. This gives rise to high costs. Moreover, representations for unseen nodes or entirely new graphs cannot be quickly generated.

Hamilton et al. [8] proposed a general framework, called GraphSAGE (SAmple and aggregaGatE), for inductive node embedding. Instead of training a distinct embedding vector for each node, GraphSAGE trains a set of aggregator functions and generates node embeddings by applying the learned aggregation functions. Based on GraphSAGE ideas, Zhou et al. [10] attempted to learn node deeper representation scores in CDGs for key class identification tasks. Besides, in the proposed method, the authors also considered the influence of the direction and weight of the edges in the Class Dependency Network (CDN).

Inspired by the above-mentioned studies and by considering the direction and weight of edges for nodes’ feature vector learning, we also attempted to apply an inductive GNN model to CDN and then used them for defect prediction. The main contributions are summarized as follows:

DOI Reference Number: 10.18293/SEKE23-068
We introduce an inductive GNN model, called GraphSAGE, to learn the features of CDN nodes effectively.

We proposed a new method named GSAGE2defect, which uses the network embedding technique node2vec to initialize the node attributes of CDNs, and uses the GraphSAGE model to further implement feature extraction to improve defect prediction.

We validated the effectiveness of GSAGE2defect based on eight open-source projects, and the results indicated that the proposed method can improve defect prediction performances.

The remainder of this paper is organized as follows: The related work is introduced in Section 2. The proposed method is detailed in Section 3. The experimental setup and results analysis are presented in Section 4. The advantages and shortcomings of our work are discussed in Section 5. Finally, the conclusion and prospects are drawn in Section 6.

II. RELATED WORK

In recent years, deep learning has been utilized to mine nonlinear features in software source code, where capturing semantic information from Abstract Syntax Tree (AST) has attracted widespread attention. For example, Wang et al. [11] extracted the source code of AST, and then leveraged Deep Belief Network (DBN) to automatically learn the hidden semantic and syntactic features in the program for defect prediction. Li et al. [12] employed a Convolutional Neural Network (CNN) to extract the semantic information of ASTs and combined the learned features with traditional hand-crafted features to enhance the prediction performance. However, AST only encapsulates the abstract syntax structure of the source code and cannot represent the execution process of the program. Hence, Phan et al. [13] converted the source code into a program Control Flow Graphs (CFG), and tried to learn from CFG through the convolutional neural network. It is worth mentioning that AST and CFG only focus on the semantic and structural information inside each code file, thereby ignoring the macro-structural information between code files, such as the dependencies between classes. Thus, Qu et al. [2] adopted a network embedding technique (i.e., node2vec) to automatically learn the external structural information of the CDN. In this way, they could achieve good results.

With the development of GNNs, the graph embedding model can integrate node and edge attributes while learning network structure. Indeed in [3], the author enhanced the performance of software defect prediction by successfully managing to learn the network structural features of source codes by using a transductive graph convolution neural network (GCN) model. Nevertheless, software system is constantly evolving in the real world. The addition of new classes, the deletion of irrelevant classes, the update of functional classes, and the repair of error classes will give rise to the iterative evolution of software systems. The transductive graph neural network (e.g., GCN) needs to re-learn the entire graph when generating embeddings for new nodes in the graph, which will result in problems such as excessive computing costs and large space costs. Besides, because network dependence is usually directed and weighted, CDG should be a directed weighted graph. However, the authors in [3,4] regarded CDG as an undirected unweighted graph. Some network information is inevitably lost in this approach.

It is not difficult to find that most existing studies have the following limitations: (1) The focus is on embedding nodes from a single fixed graph, while many real-world applications require the fast generation of embeddings for unseen nodes or entirely new (sub) graphs. (2) The weight and direction of the edges are not considered during software network modeling. In practice, the dependency between classes is not a bidirectional equivalent dependency, and the degree of dependency between classes also varies.

Given these, we introduced an inductive graph neural network, namely GraphSAGE, to learn the directed weighted CDN. Specifically, we constructed a GSAGE2defect model for learning the structural features of nodes in class dependency networks. Then we performed end-to-end learning based on the semantic and structural information of nodes, and ultimately applied the obtained features to improve defect predictive performance.

III. METHOD

The framework diagram of this research is shown in Figure 1, which mainly includes three parts: (1) Constructing a class dependency network, and then learning the node embedding by the node2vec method; (2) Initializing node attributes by combining learned node embedding and traditional hand-crafted metrics, then introducing GraphSAGE to learn network structural features; (3) Training a classifier for defect prediction.

A. Network Modeling and Embedding Learning

1) Class dependency network modeling

A Class Dependency Network (CDN) is a directed weighted network constructed according to the dependencies between class files. For object-oriented software, its class dependency network $CDN_p = (V, E)$, where $V$ is the set of nodes, each node $v \in V$ represents a class or interface, $E$ is the set of edges, representing the dependencies between classes or interfaces. Three main dependencies are considered in this paper: Inheritance dependency, Interface implementation dependency, Method calls dependency (aggregation).

For weight extraction, the calculation method of the edge weight was as follows:

$$W_{ij} = \frac{d_{ij}}{\sum_{k \in N(j)} d_{ik}}$$

where $W_{ij}$ represents the weight between node $v_i$ and node $v_j$, $d_{ij}$ stands for the number of dependencies between the two nodes, and $N(j)$ denotes the set of neighbors of the node $v_j$.

It is worth mentioning that in the weighted network, the weight of the edge is not related to the direction, but to the current target node. Some explanations are mentioned in reference [10].

2) Node attribute generation
Before training GraphSAGE, we must provide the attributes of the nodes in CDN. Node attribute metrics can include many types, such as traditional static code metrics, complex network metrics, and network-embedded metrics. Traditional static Code Metrics (TCM) consist of twenty manually designed metrics, such as CBO (number of classes coupled to a given class), WMC (number of methods in a given class), and LOC (lines of source code). Complex Network Metrics (CNM) are extensively employed in social networks, including seventeen metrics such as density, size, and extent. Network structure information is extracted from CDN through network embedding learning as a Network Embedding Metric (NEM). We used the node2vec [14] method to map each class node to a low-dimensional vector.

Different metrics can be combined in many ways. According to our previous research [4], the combination of TCM and NEM as node attributes is the optimal choice. Therefore, we selected this combination of metrics as the initial attributes of nodes in class dependency networks to feed into GraphSAGE for training.

B. GraphSAGE learning

Algorithm 1 describes the entire learning process. Specifically, a CDN and its node attributes are provided as input. Each step in the outer loop of Algorithm 1 proceeds as follows, where k signifies the current search depth in the outer loop, and \( \mathbf{h}_v^k \) indicates the node representation at this search depth. First, each node \( v \in V \) aggregates the representations of the nodes in its immediate neighborhood, \( \mathbf{h}_{u}^{k-1}, \forall u \in N(v) \), and then convert to a single vector \( \mathbf{h}_v^{k-1} \). After aggregating adjacent feature vectors, GraphSAGE concatenates the current representation of the node \( \mathbf{h}_v^{k-1} \) with the aggregated domain vector \( \mathbf{h}_N^{k-1} \) to obtain the features of the current layer node representation, the final output is the feature matrix \( \mathbf{Z} \) of the node.

Algorithm 1: GraphSAGE algorithm for generating node features

<table>
<thead>
<tr>
<th>Input:</th>
<th>( G(V, E) ); Node initial feature vector ( \mathbf{X} ); Depth ( K ); weight matrix ( \mathbf{W}_k, \forall k \in {1, \ldots, K} ); aggregation functions ( AGGREGATE_k, \forall k \in {1, \ldots, K} ); neighborhood function ( N: V \rightarrow 2^V )</th>
</tr>
</thead>
</table>
| Output: | Node Feature Matrix \( \mathbf{Z} \).
1 Initialize the node representation vector \( \mathbf{h}_v^0 = \mathbf{X}_v, \forall v \in V \);
2 if \( k = 1 \ldots K: \)
3 do the following for each node \( v \in V \):
4 \( \mathbf{h}_{N(v)}^k = AGGREGATE_k((\mathbf{h}_u^{k-1}, \forall u \in N(v))) \);
5 \( \mathbf{h}_v^k = \sigma (\mathbf{W}_k \cdot \text{CONCAT}(\mathbf{h}_{N(v)}^k, \mathbf{h}_v^{k-1})) \);
6 \( \mathbf{h}_v^k = \mathbf{h}_v^k / \| \mathbf{h}_v^k \|_2 \); // Normalize the \( \mathbf{h}_v^k \) obtained by each layer
7 ends
8 \( \mathbf{z}_v = \mathbf{h}_v^k, \forall v \in V \);
9 Output node feature matrix \( \mathbf{Z} \).

GraphSAGE provides three aggregator functions: average aggregation, Long Short-Term Memory (LSTM) aggregation, and max pooling aggregation. we selected the default aggregator function (i.e., max pooling aggregation).

IV. Experiment

A. Datasets

In our experiments, we utilized eight classical defect projects published by the PROMISE\textsuperscript{1} library. Table I shows the details of the eight software projects, in which #Nodes represents the number of class files in CDN, #Edges indicates the number of dependencies between class files, #Defective indicates the number of buggy files in the project, and %Defective denotes the corresponding buggy rate.
B. Experimental setup

Because the defect dataset was imbalanced, it needs to be sampled before training. Here we consider 4 commonly used sampling strategies: SMOTE [15], BorderlineSMOTE [16], SMOTETENN [17] and SMOTETomek [18].

Five-fold cross-validation was used (the training set accounted for 80% and the test set 20%), and we repeated the experiment 25 times. The final results were averaged to reduce the bias introduced by randomly dividing the data. The experimental parameters of this study are shown in Table II.

TABLE I. DATASET

<table>
<thead>
<tr>
<th>project</th>
<th>version</th>
<th>#Nodes</th>
<th>#Edges</th>
<th>#Defective</th>
<th>%Defective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant</td>
<td>1.7.0</td>
<td>703</td>
<td>3012</td>
<td>128</td>
<td>22.76%</td>
</tr>
<tr>
<td>Camel</td>
<td>1.6.0</td>
<td>906</td>
<td>3644</td>
<td>145</td>
<td>20.09%</td>
</tr>
<tr>
<td>Ivy</td>
<td>2.0</td>
<td>343</td>
<td>1710</td>
<td>31</td>
<td>11.37%</td>
</tr>
<tr>
<td>JEdit</td>
<td>4.1</td>
<td>292</td>
<td>1044</td>
<td>58</td>
<td>25.00%</td>
</tr>
<tr>
<td>Velocity</td>
<td>1.6.1</td>
<td>210</td>
<td>1035</td>
<td>60</td>
<td>35.71%</td>
</tr>
<tr>
<td>Poi</td>
<td>3.0</td>
<td>421</td>
<td>1304</td>
<td>273</td>
<td>64.85%</td>
</tr>
<tr>
<td>Lucene</td>
<td>2.4.0</td>
<td>324</td>
<td>1353</td>
<td>194</td>
<td>59.88%</td>
</tr>
<tr>
<td>Xalan</td>
<td>2.6.0</td>
<td>801</td>
<td>3965</td>
<td>362</td>
<td>45.19%</td>
</tr>
</tbody>
</table>

TABLE II. EXPERIMENTAL ENVIRONMENT

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature dimension</td>
<td>32</td>
</tr>
<tr>
<td>Epochs</td>
<td>2000</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Initial learning rate</td>
<td>0.001</td>
</tr>
<tr>
<td>Output dimension</td>
<td>32</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1</td>
</tr>
<tr>
<td>Imbalance Handling Threshold $\sigma$</td>
<td>0.4</td>
</tr>
</tbody>
</table>

To verify the effectiveness of GSAGE2defect, we selected the following eight benchmark methods for comparison: dw2defect [18], node2defect [2], DP-CNN [12], Seml [19], SDNE2defect [20], Struc2defect [21], GCN2defect [3], and GAT2defect [22].

C. Evaluation metrics

The evaluation metric of our experiment adopts F-measure. The formula for calculating F-measure is as follows:

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Among them, TP (True Positive) and TN (True Negative) are the numbers of positive and negative samples predicted correctly, FP (False Positive) is the number of negative samples predicted as positive samples, and FN (False Negative) is the number of positive samples predicted as negative samples.

D. Results and Analysis

**RQ1: Does the proposed GSAGE2defect approach work well?**

For a fair comparison, the sampling method and classifier in the benchmark model were chosen consistent with our proposed method. All baseline models use SMOTETomek and Random Forest (RF) in predictions.

To further explore RQ1, we contrasted our approach with four baseline models and four classical model approaches. Figure 2 (left) exhibits that our proposed method is superior to the four baseline methods in terms of overall performance, which is evident by the higher median value of the F-measure (0.919). Among the four baseline methods, the performance of the GAT prediction model of the graph attention neural network (labeled as GAT2defect) was better than the other three baseline methods, and its median value of the F-measure was 0.843. Moreover, Figure 2 (right) displays that the proposed method is also better than the four classical methods. This is reflected by its median value of the F-measure which is the greatest (0.919).

The overall F-measure value of dw2defect was 0.793, struc2defect was 0.702, sdne2defect was 0.804, GAT2defect was 0.843, and GSAGE2defect was 0.917. The performance of GSAGE2defect was indeed better than the four baseline methods. The F-measure values of the other four classical methods of DP-CNN [12], Seml [19], node2defect [2], and GCN2defect [3] were 0.807, 0.714, 0.714, and 0.894, respectively. The performance of GSAGE2defect was clearly better than the four classical methods.

Figure 3 (left) depicts the improvement in the prediction of the GSAGE2defect model compared to that of the eight benchmark models. It can be observed from the figure that the GSAGE2defect model has provided a certain improvement in the prediction compared with other benchmark models, and the improvement range is between 2.09% and 26.69%.

In general, according to the results of Figure 3, it can be found that the GSAGE2defect model has a better improvement than the benchmark model, with a maximum improvement of 26.69% and an average improvement of 13.22%.

**RQ2: Do the dependency direction and weight of CDN have a significant impact on GSAGE2defect?**

To answer RQ2, we designed four sets of experiments to discuss the prediction performance of the GSAGE2defect model in four network scenarios and to explore the influence of the dependency direction and weight on the model during network modeling. The results are displayed in Figure 3(right) and Table III. Table III shows the F-score of the GSAGE2defect model under eight projects in four network scenarios.

On the whole, the GSAGE2defect model had the best performance under the directed and unweighted network with the highest average F-score (0.921), and five of the eight projects achieved the highest F1-score in the directed and unweighted scenario, followed by the scenario of undirected weighted and directed weighted, and the worst was undirected unweighted.

Besides, we introduced Wilcoxon signed-rank test (p-value) and Cliff’s [23] influence factors, and compared and analyzed the differences in the experimental results. According to the results of the p-value and d-value in Table 4, the class dependency direction and weight do not have a very substantial impact on the prediction performance of the GSAGE2defect model.
TABLE III. COMPARISON OF MICRO-F1 RESULTS IN FOUR NETWORK SCENARIOS

<table>
<thead>
<tr>
<th>Project</th>
<th>ud &amp; uw</th>
<th>ud &amp; w</th>
<th>d &amp; uw</th>
<th>d &amp; w</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant</td>
<td>0.907</td>
<td>0.908</td>
<td>0.903</td>
<td>0.905</td>
</tr>
<tr>
<td>Camel</td>
<td>0.936</td>
<td>0.941</td>
<td>0.937</td>
<td>0.937</td>
</tr>
<tr>
<td>Ivy</td>
<td>0.937</td>
<td>0.938</td>
<td>0.937</td>
<td>0.939</td>
</tr>
<tr>
<td>jEdit</td>
<td>0.919</td>
<td>0.923</td>
<td>0.932</td>
<td>0.925</td>
</tr>
<tr>
<td>Velocity</td>
<td>0.913</td>
<td>0.918</td>
<td>0.918</td>
<td>0.911</td>
</tr>
<tr>
<td>Poi</td>
<td>0.920</td>
<td>0.928</td>
<td>0.930</td>
<td>0.928</td>
</tr>
<tr>
<td>Lucene</td>
<td>0.919</td>
<td>0.919</td>
<td>0.921</td>
<td>0.921</td>
</tr>
<tr>
<td>Xalan</td>
<td>0.888</td>
<td>0.881</td>
<td>0.892</td>
<td>0.887</td>
</tr>
<tr>
<td>Average</td>
<td>0.917</td>
<td>0.920</td>
<td>0.921</td>
<td>0.920</td>
</tr>
</tbody>
</table>

comparison Sig, p < 0.05, d
ud&uw vs ud&w 0.1913 (-0.1875)
d&w vs d&uw 0.1627 (0.0625)
u&d&uw vs ud&w 0.0780 (-0.2180)
u&d&uw vs d&w 0.8321 (0)

V. DISCUSSION

For addressing RQ1, we reproduced the GCN2defect method on the dataset processed in this paper. The defect rate of the dataset in our study was slightly different from that in [3]. This may originate from the fact that the direction information and weight information between nodes were not extracted in [3] when processing the dataset, however, we extracted these two types of information.

For addressing RQ2, the comparison of directed unweighted vs. directed weighted and undirected weighted vs. directed weighted groups demonstrated that considering the effect of both direction and weight is not optimal compared to considering a single element. In the comparison between undirected unweighted and undirected weighted cases, the F-score of seven out of eight projects in the undirected weighted was greater than the former and the d-value was negative. This indicated that in the undirected case, the weighting effect was better. Compared with the directed and unweighted case, the F-score of six out of the eight projects of the directed and unweighted case was greater than the undirected unweighted case and the d-value was negative, suggesting that the directed effect was better in the case of the unweighted. Therefore, it is still necessary to consider the dependency direction and weight in the software network modeling process.

Moreover, we chose random forest as the classifier in RQs. To further explore the influence of the classifier on the model predictions, we tried four different classifiers in the GSAGE2defect model, including Random Forest (RF), Multilayer Perceptron (MLP), Decision Tree (DT), and Logistic Regression (LR). The GAGE2defect model worked best in directed and unweighted scenarios. Thus, in this scenario, we conducted a comparative experiment of classifiers in the GSAGE2defect model. Table IV exhibits the effects of the four classifiers on GSAGE2defect, and the best values are shown in bold. In addition, Table V shows the significant differences when using different classifiers. From the results of the p-value and d-value, it can be observed that the difference between RF and the other three classifiers is significant.

TABLE IV. THE EFFECT OF THE CLASSIFIER ON GSAGE2DEFECT

<table>
<thead>
<tr>
<th>Project</th>
<th>LR</th>
<th>DT</th>
<th>MLP</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant</td>
<td>0.732</td>
<td>0.653</td>
<td>0.830</td>
<td>0.903</td>
</tr>
<tr>
<td>Camel</td>
<td>0.805</td>
<td>0.757</td>
<td>0.872</td>
<td>0.937</td>
</tr>
<tr>
<td>Ivy</td>
<td>0.800</td>
<td>0.689</td>
<td>0.812</td>
<td>0.937</td>
</tr>
<tr>
<td>jEdit</td>
<td>0.821</td>
<td>0.781</td>
<td>0.833</td>
<td>0.932</td>
</tr>
<tr>
<td>Velocity</td>
<td>0.827</td>
<td>0.818</td>
<td>0.320</td>
<td>0.918</td>
</tr>
<tr>
<td>Poi</td>
<td>0.922</td>
<td>0.907</td>
<td>0.460</td>
<td>0.930</td>
</tr>
</tbody>
</table>
Lucene  0.892  0.854  0.450  0.921
Xalan  0.893  0.823  0.872  0.892
average  0.836  0.785  0.681  0.921

TABLE V. STATISTICAL TEST RESULTS OF GSAGE2defect

<table>
<thead>
<tr>
<th>GSAGE2defect</th>
<th>p-value</th>
<th>Cliff’s delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR vs RF</td>
<td>0.016</td>
<td>-0.288</td>
</tr>
<tr>
<td>DT vs RF</td>
<td>0.006</td>
<td>-0.938</td>
</tr>
<tr>
<td>MLP vs RF</td>
<td>0.019</td>
<td>-0.938</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

This paper proposed a new defect prediction method GSAGE2defect, by introducing an inductive graph neural network model GraphSAGE to automatically learns the dependencies between nodes in a class dependency network and using SMOTE-Tomek sampling to solve the problem of sample imbalance. We verified the effectiveness of our method on data from eight open-source projects, and the results indicated that GSAGE2defect can outperform the baseline model by 26.69% in terms of F-measure, and also revealed that considering weight and direction in the class dependency network is helpful for software defect prediction. If only one of these two factors is considered, a certain improvement will occur in the predictions of the model. In general, the model had the best prediction in the directed and unweighted scenario. Besides, our method has an absolute advantage compared to the four baseline models, and it also has obvious advantages in comparison to the five classic methods.

This study is only a small part of the results of our comprehensive research, and there is still much research that should be completed. In the future, we will extend this study to extract richer features and to build more complete software networks. Furthermore, we will extend our research to focus on defect prediction across various projects.

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

Small Educational Steps Towards Improving the Status of Women in Software Engineering

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

Nazlie Shahmir  
Information Services  
Canadian Pacific Railway Company  
Calgary, Canada  
nazlie_shahmir@cpr.ca

Abstract—The women in software engineering continue to face a culture of discord that manifests itself in the form of underrepresentation, unpleasantness, and/or inequitableness. This somewhat dire situation was only exacerbated during the COVID-19 pandemic when the women in software engineering education and profession had to deal with multiple ‘crisis’. The status quo is clearly unacceptable, not least because of pervasiveness of software in society. In that regard, relying on a multipronged approach grounded in a body of knowledge, ethics, and history, this paper proposes certain basic steps in software engineering courses and projects that could be put into practice for improving “gender literacy” among students. These educational steps are illustrated by anecdotes and examples.

Keywords—accessibility; community smell; DEI; history; persuasion; SECEPP; social activism; SWEBOK

I. INTRODUCTION

It is commonly understood that, to a large extent, the engineering of software is inherently a social endeavor. Then, creating an amicable and amenable social environment that maximizes the potential of all stakeholders involved, regardless of their gender, is therefore natural. However, when it comes to inconsideration, the computing community, in general, and the software engineering community, in particular, has a long disreputation of singling-out one gender, namely women [1–3]. For the rest of the paper, SoWSE is used as an all-encompassing term to denote the current status of women in relation to software engineering.

This paper employs a multipronged preventative approach to addressing SoWSE early, that is, during the phase in the lives of men and women when they are students and therefore still learning and growing, in the hope that attitudes and/or behaviors developed and shaped early during their software engineering education (SEE) will carry over later during their software engineering profession. The steps involved are based on authors’ own educational and professional experiences of more than 25 years, derived from “first principles”, aligned with previous work, and require minor (re)orientation in the current direction of SEE. These relatively smaller steps can serve as a basis as well as a prerequisite for a commitment to relatively larger steps. The issues pertaining to non-binary genders, albeit important, are beyond the scope of this paper.

The rest of the paper is organized as follows. In Section II, necessary background is provided and related work is discussed in order to set the stage of the underlying problem.

II. BACKGROUND AND RELATED WORK

A. A Glimpse into the Nature of a Perennial Social Problem and Proposed Solutions

1) An Epigrammatic Synthesis of the Problem

The women in software engineering (WSE) have been under siege for the past several decades, spanning 20th and 21st centuries. The social issues continue to be faced by WSE in educational and/or professional settings include the following: dissonance (for example, comments or questions not taken seriously) [4], empowerment imbalance (for example, low recognition or low reward) [4], harassment (for example, name-calling on social media) [5], inadequate support (for example, lack of growth opportunities or inflexibility during the COVID-19 pandemic) [6–8], lack of autonomous decision-making (for example, mandated supervision on significant matters) [7], lack of peer parity (for example, absence of peers, mentors, or role models) [4], marginalization (for example, not invited or discouraged to participate in software project team meetings) [7], maternal discrimination (for example, overtly or covertly discouraged to continue after maternity leave) [7], microaggressions (for example, exoticization of women of ethnic minority during interactions), ostracization and isolation (for example, by being the only women in a software project team, sometimes repeatedly) [4, 9], salary imbalance (for example, compared to men in the same or similar positions) [7], scapegoating (for example, ridiculed and blamed, not always with due diligence, for shortcomings) [7, 9], stereotyping (for example, told implicitly or explicitly that men are better suited) [4], and workload imbalance (for example, obligated to overwork) [6–8].

These issues can have a long-term impact on mental and/or physical health of those women who are compelled to adapt and need to resort to coping and mitigation strategies in order to continue to be involved in some capacity in software engineering education or profession [9], and even of those who do not. They also hardly engender a natural environment for essentially any type of work for women, let alone a productive environment for engineering of software, not to mention that resilience and compromise have their inevitable limits.
The recent large-scale movements initiated, mobilized, and sustained by women, such as #metoo and #mahsaamini, only reinforce that the breadth and depth of the problem is not specific to software engineering, or even computing.

2) An Epigrammatic Synthesis of the Solutions

In the past few decades, multiple solutions have been proposed, at different times, examining SoWSE in different geographical regions, at different breadths and depths, from different perspectives, and published in different avenues. For example, motivational (but, sometimes, opportunistic and self-serving) speeches to girls and women have been given [4]; prevalence of a “broprogrammer culture” has been raised [4, 10]; labels such as “conscious and subconscious) gender bias”, “gender disparity”, “gender divide”, “gender inequality”, and “gender inequity” have been created [11]; recurring pseudo-scientific claims about the “predispositions” and “differences” between men and women have been debunked and clarified [12]; myths and misconceptions of “gender-neutrality” of software have been discarded and clarified [13]; surveys to determine root causes of the plight of WSE have been conducted [11]; calls for action have been made [11]; proposals have been put forth [11]; slogans have been proselytized [11]; and books have been written [14]. Yet, not much has changed over the generations and the struggle of WSE continues unabated today.

It is possible that the problem may be extremely difficult, even “wicked”, and a lasting solution elusive and unsustainable to eventalism (say, in form of “big bang”). However, it at the very least does show that there is a lack of sustained interest, perhaps even a lack of empathy, at scale, by those in positions of influence or privilege in changing the status quo for the better. This is unsurprising, as a change would require admitting of their own volition that there indeed is a problem and they are part of the problem [11], overcoming the bystander effect, and need to accept responsibility and bear a cost to self to be a part of its solution, a tall order for many.

B. Implications for Interplay between Software and Society

1) Unacceptable Status Quo of Women in Software Engineering

The consequences are solemnly evident: women are part of a culture in which they are not naturally welcome in software engineering, which implies and is implied by, there are a disproportionately low number of WSE (meaning, there are women who either do not enter software engineering or do not stay in software engineering), and there is a significant number of WSE who, ceteris paribus, are not treated the same as men in software engineering and treated as “second-class”. Fig. 1 iconizes this continuing self-fulfilling prophecy.

This is extremely disconcerting also to women in other disciplines who rely intimately and regularly on the use of software, and are therefore affected, directly or indirectly, by how software is engineered. It patently situates software engineering in a uniquely precarious and potentially disreputable position among all the engineering disciplines.

2) Unacceptable Status Quo of Software Engineering Products and Services for Women

The aforementioned consequences inevitably also have a lasting ripple effect on both open-source and closed-source software engineering, and manifests itself in different ways: rejection (a large segment of society is disregarded in educating prospective software engineers or employing qualified software engineers), suppression (women with sustained interested in software engineering do not have the opportunities to reach their full potential), elimination (the retention rate of women is threatened unnecessarily), and exclusion (the needs and preferences of a large segment of potential users are not taken into account in developing and/or maintaining socio-technical interactive systems). In other words, women are essentially not welcome either as producers or as consumers of software. The message, implicitly or explicitly, seems to be that women and software should exist and persist in two completely separate worlds, as implied by Fig. 1, where one never comes across the other, an evident implausibility in today’s society, to say the least.

The systems resulting from such ‘engineering’ (as a euphemism for ‘persecution’) cannot be considered as truly ‘representative’. This, in turn, means that such systems are, by design, not “gender-neutral”, and any assertions of such systems being utilisable and usable by women are speculative. In addition, such non-representativeness does not get any better if the designers of the aforementioned systems are “asshole designers” [5, 15] creating “asshole design” (defined as “design that possesses clearly coercive, deceptive, and/or malicious intent on the part of its creator, rather than be inconvenient to a user due to a poor design decision or implementation” [15]).

The indelible loss of those quality attributes deemed necessary for socio-technical interactive systems is not only a matter of grave concern in and of itself, but even more so given the increasingly indispensable role being played by software in society-at-large. Indeed, such reliance-by-necessity on software was only accentuated and reaffirmed repeatedly during the mandated lockdowns at the peak of the COVID-19 pandemic.

C. SEE and SoWSE

SEE has a long and illustrious history [16, 17], and undergraduate and/or graduate software engineering programs continue to proliferate globally. However, the issues pertaining to gender have to date been discussed rarely and peripherally [18], which is one of the motivations for this paper.

III. A Compendium of Small Steps in Software Engineering Education as Part of the Solution

A. Improving SoWSE and Software Engineering Courses

There are a number of concrete steps that can be taken, both inside and outside the classroom, towards improving SoWSE.
From an epistemological perspective, these steps advocate ethically-sensitive and historically-guided SEE, apply to both undergraduate and graduate students, represent iterative and incremental rather than radical changes in attitudes and/or behaviors on part of teachers and students, ‘nudging’ the students slowly but certainly purposively by means of positive as well as negative reinforcements towards desired outcomes [19]. (These steps are part of a marathon, not a sprint.)

From a logistical perspective, these steps are inexpensive in terms of time and effort, and do not require acquisition or management of extra resources by the institutions. Indeed, as seen later, they could be integrated naturally in a variety of software engineering-related courses.

1) Improving SoWSE and Software Engineering Outreach

The Web Sites of courses, such as those using Moodle or another learning management system, or even otherwise, could be used as venues for creating awareness among students of events, movements, and organizations that (do not subscribe to #genderwashing but) are aimed towards accentuating and celebrating the accomplishments of women in computing, in general, and software engineering, in particular. Fig. 2 shows such an example.

![Figure 2. A montage of initiatives committed to congregating, recognizing, and supporting WSE.](image)

Furthermore, the teachers could, for example, invite guest women speakers on the International Women in Engineering Day or on other auspicious occasions, and the students, both men and women, could be encouraged (and, if possible, even credited) to participate, volunteer, or work, as suitable, for these outlets. (The authors, incidentally, have been involved with Chic Geek and are signatories of the Diversity Charter.) Indeed, the broad availability both inside and outside the classroom of videoconferencing tools, such as Microsoft Teams and Zoom, has to a large extent overcome the logistical limitations due to time and place, and made such speaker arrangement and participation ever so easier.

2) Improving SoWSE and Software Engineering Ethics

The Guide to the Software Engineering Body of Knowledge (SWEBOK) represents the current state of generally-accepted consensus-based knowledge emanating from interplay between theory and practice of software engineering [20]. The SWEBOK is decomposed into a set of topics through Knowledge Areas (KAs). The Software Engineering Professional Practice KA of SWEBOK is especially relevant to this paper as it mentions codes of ethics.

The ACM/IEEE Software Engineering Code of Ethics and Professional Practice (SECEPP) lists generic principles, each of which is refined into specific clauses, related to the behavior of and decisions made by professional software engineers as well as students of the profession.

In particular, the following clauses from Principle 5: MANAGEMENT, Principle 7: COLLEAGUES, and Principle 8: SELF, respectively, are applicable:

- 5.07. Offer fair and just remuneration.
- 5.08. Not unjustly prevent someone from taking a position for which that person is suitably qualified.
- 7.04. Review the work of others in an objective, candid, and properly-documented way.
- 7.05. Give a fair hearing to the opinions, concerns, or complaints of a colleague.
- 8.07 Not give unfair treatment to anyone because of any irrelevant prejudices.

In teaching a course involving any of the aforementioned SECEPP clauses, or even otherwise, the students could be introduced to at least those fallacies and biases that can impede rational thinking and are team dynamic-specific or gender-specific. For example, these could include, but are not limited to, the following: Argumentum Ad Hominem, Appeal to Authority, Appeal to Flattery, and Cherry Picking (logical fallacies), the Bias Blind Spot and the Dunning-Kruger Effect (meta-cognitive biases), and the Gender Stereotyping Bias, the Confirmation Bias, the Affinity Bias, and the Inter-Group Bias (cognitive biases). The students could, time permitted, also be introduced to refutation and debiasing techniques.

3) Improving SoWSE and Software Engineering History

There have been several notable contributions by women to computing since the dawn of computing [21, 22, 23]. Indeed, the indispensable role of women in computing, in general, and cost estimation, in particular, was evident even more than 75 years ago [1]:

Sometime in 1944, computers became “girls.” The University of Pennsylvania hired “girl computers”; Warren Weaver started calling Applied Mathematics Panel computers “girls”; Oswald Veblen, who had once led a team of computing men, used the term “girls”; George Stibitz began ranking calculating projects in “girl-years” of effort.

Furthermore, the term ‘software engineering’ was coined in the early 1960s by Margaret Hamilton. However, the ‘standard’ software engineering textbooks seldom address the human or social aspects of the history of software engineering at any depth, and rarely highlight the role of women per se in advancing software engineering to its current state.

In teaching a course based on any of the SWEBOK KAs, or even otherwise, the students could be introduced to the genealogy of that KA, moving both backward and forward in time, where contributions of women alongside men (and the challenges that they overcame) could be highlighted. For example, Table 1 shows areas of contributions of certain women in one KA from SWEBOK, Version 3 and another KA...
from SWEBOK, Version 4. (The presence of three entries in each case is intentional and follows the ‘Rule of Three’ criterion of a (software) pattern.)

### TABLE I. EXAMPLES OF WOMEN CONTRIBUTORS TO SWEBOK KAS

<table>
<thead>
<tr>
<th>SWEBOK KA</th>
<th>Notable Women Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Architecture</td>
<td>Patricia Lago, Barbara Paech, Mary Shaw</td>
</tr>
<tr>
<td>Software Design</td>
<td>Nancy Leveson, Perdita Stevens, Rebecca Wirfs-Brock</td>
</tr>
</tbody>
</table>

Indeed, such efforts need not be restricted to SWEBOK KAs, and could extend to cognate disciplines. For example, Table 2 shows areas of contributions of certain men and women in human-computer interaction (HCI).

### TABLE II. EXAMPLES OF MEN AND WOMEN CONTRIBUTORS TO HCI

<table>
<thead>
<tr>
<th>HCI Area</th>
<th>Notable Men and Women Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction Design</td>
<td>Helen Sharp, Ben Shneiderman, Jennifer Tidwell</td>
</tr>
<tr>
<td>Mental Models</td>
<td>Felienne Hermans, Donald Norman, Indi Young</td>
</tr>
</tbody>
</table>

The publications, recordings of speeches, or other work by women software engineers could serve, for example, as topics for classroom discussions and/or presentations, as reading material for term papers, and as a basis for essay-type, open-ended questions on, say, assignments and/or examinations.

#### B. Improving SoWSE and Software Engineering Projects

It is common for software engineering-related courses to have semester-long team-based projects as a major component of the overall assessment. These projects tend to have several goals, including developing interpersonal skills for working in a team environment in the software industry and beyond [24].

1) **Improving SoWSE and Software Project Topics**

The teachers could give software project topics that are appealing to both men and women students. For example, this could be ascertained by conducting an informal poll at the beginning of the course. Indeed, there are several systems today, including Google Forms and Moodle, which provide effective and efficient means to conduct such polls.

2) **Improving SoWSE and Software Project Teams**

There are at least two common ways in which teams for projects are formed, each with their own advantages and disadvantages [24]: (a) either, based on some criteria, the teachers place the students in teams, or (b) the students could be asked to form their own teams. In either case, every member in each team needs to play a role, such as a team leader, modeler, designer, programmer, tester, documenter, and so on.

The teachers could ensure that each role is considered equally important and receives equal credit, and that the roles assigned within each team rotate during the semester (to avoid compartmentalization and for each member to gain experience in each role, and to avoid stereotyping of any particular gender to any particular role). Furthermore, to reduce the effect of the so-called Imposter Syndrome [25], women could be encouraged to “experiment”, meaning to try and make mistakes early and, in some cases, deliberately in the process [26], so that they not only learn to make less of the same or similar types of mistakes later, but also develop their confidence and competence in their adopted role over time. Indeed, doing so is among the characteristics of agile methodologies, design thinking, and prototyping. Finally, each member could be asked to create and maintain throughout the duration of the project a private learning journal chronicking, for example, intrapersonal and interpersonal positive and/or negative experiences, challenges encountered along the way and means used to overcome them, and signs of progress made. The contents of such a journal could be shared selectively with the others later if deemed useful for reflection and retrospective.

3) **Improving SoWSE and Software Process**

The students could be asked to adopt a software development methodology that has a low probability of the presence of community smells (such as the Organizational Silo Effect or Radio Silence) [27] and, therefore, accrual of the so-called Social Debt [27]. This is because, according to the Conway’s Law, the type of the software development methodology can determine the communication structure of a software project team, irrespective of gender. In this regard, human-centered agile methodologies have been relatively more successful than machine-centered rigid methodologies, due to their inherent support for communication and collaboration in carrying out stated activities and creating corresponding artifacts. The selected agile methodology could be customized to be ethically-sensitive and gender-inclusive [28], using the SECEPP, the ISO/IEC/IEEE 24748-7000:2022 Standard, or even otherwise.

The software development process adopted could include practices, such as collective brainstorming and mind mapping [29, 30], which necessitate interaction among all team members, including women, to learn about and learn from each other. Furthermore, the software project teams could be asked to use a Social Wiki for communication and collaboration [31]. The postings on the Wiki could be subject to an enforceable Communications Protocol that includes policies for reportable conduct related to civility and etiquette, among other things, and could be monitored by teaching assistants (TAs). (The argument generalizes to other social media technologies and tools, and more generally to the use of the Social Web.)

4) **Improving SoWSE and Software Product**

In recent years, there has been an increasing awareness of having diversity, equity, and inclusivity (DEI) [32]. By embracing DEI and being more supportive of women, the teachers can show that they are “moving with the times”, they “do agile” but also “be agile”, and that “women software engineers matter”, thereby making courses more appealing to women and contribute to better controlling attrition rate.

For example, as part of software requirements elicitation for socio-technical interactive systems, the interviews could include women interviewees and user models could include empathy map-based women personas [33]. (The argument extends to other dimensions of intersectionality, such as women of ethnic minority and women with mobile disabilities.) For the development of such systems, DEI could be among non-functional (specifically, quality) requirements [13], to ensure its consideration during subsequent stages of software development. Finally, women could be encouraged to get involved in deciding the acceptance criteria, envisaging interaction design, and/or conducting user acceptance testing.
C. Other Steps for Improving SoWSE

1) Improving SoWSE and Maternity

It is not uncommon these days to come across returning students, especially graduate students but even undergraduate students pursuing, say, another degree, having their own family already or planning to start their own family.

The teachers could be sensitive towards students who are pregnant or have very young children that require constant care, and proactively reach out and find ways to accommodate them, as necessary, while still being fair to the other students, even when such accommodation may not be mandated by their institution. (This is not all that different from accommodating those students who have identified themselves as having one or more disabilities. The argument extents to paternity.)

2) Improving SoWSE and Teaching Assistants

For courses with large class sizes, it is common to have multiple TAs for support. The TAs are expected to be in touch with the students in unique ways, serve as candidates for role models, even mentors, and can play an important role in creating a congenial atmosphere in which the software project teams can operate and the software projects can evolve.

The teachers could support women students by encouraging them to apply for the positions of TAs, and by ensuring that they acknowledge the contributions of these TAs, giving them due credit, as appropriate, while avoiding benevolent sexism, say, by appearing unnecessarily charitable. For example, this could be done by at least having a transparent criteria for recommending students for the positions of TAs before the beginning of the course, and sending a personalized ‘Thank You’ note for their help after the end of the course. The teachers could also guide, as necessary, men TAs to work harmoniously with women TAs. For example, this could be done by making expectations regarding gender inclusivity explicit and developing A Guide for Teaching Assistants that the TAs need to abide by for continued employment.

D. Limitations

It is implicit throughout the aforementioned steps that the teachers themselves are prepared to deal with the issues pertaining to gender, in general, and women, in particular. In certain institutions, there are seminars and workshops related to DEI, but discussion of these is beyond the scope of this paper.

IV. DIRECTIONS FOR FUTURE RESEARCH

There are a number of directions that emanate naturally from this paper.

A. Looking Forward to Change: COVID-19 and SoWSE

The COVID-19 pandemic has been an unprecedented, life-altering event, especially for those who became seriously ill or suffer from its long-term effects, or lost loved ones. It also has led people to self-assess and question their ingrained beliefs and values, and consequences of their actions. In that regard, it would be useful to conduct a survey of students aimed to investigate whether there has been any notable change among men in their attitudes and/or behaviors as per SoWSE before and after having been introduced to the steps proposed in this paper, their rationale for a change, the things they had to do as well as not do in making a change, and personal cost for making a change, and is therefore also of research interest.

B. Looking Forward to Change: Men and SoWSE

It is unsurprising that men have an important role to play in changing the status quo of SoWSE to an extent that could be considered acceptable by women. In that regard, it would be useful to investigate whether there has been any notable change among men in their attitudes and/or behaviors as per SoWSE before and after having been introduced to the steps proposed in this paper, their rationale for a change, the things they had to do as well as not do in making a change, and personal cost for making a change, and is therefore also of research interest.

V. CONCLUSION

I am no longer accepting the things I cannot change.
I am changing the things I cannot accept.

― Angela Davis

In the last 50 years or so, software engineering as a discipline has progressed far less socially than it has industrially and technically. This is not only unsustainable, it is antithetical to the essence of software engineering, and, by reference, to the spirit of software as a means for societal good.

The status quo of WSE is not only unnecessarily constraining and profoundly inegalitarian, the society has no compelling reason or the right to deprive women of the delights (and disappointments) and triumphs (and tribulations) of a software engineering experience. The WSE (or women in any other discipline of interest or matter of relevance) need to be treated as “first-class” in a society that vies to be progressive and prosperous. To do that at scale, as with dealing with the COVID-19 pandemic, “it will take all of us” as opposed to “everyone for themselves”. If not, then for many women software engineering has regressed to their detriment.

The engineering of software evidently involves being “polyliterate” not only in a combination of multiple technical dimensions, but also multiple non-technical dimensions. As the history and evolution of software engineering has shown, both the number of dimensions and the nature of each dimension are a function of time, and can therefore change over time. In particular, there is ever increasing attention on non-technical dimensions, especially soft skills [34]. SEE needs to evolve accordingly [35]. In that regard, it is the thesis of this paper that a software engineering-specific “gender literacy” (defined as “an awareness and understanding of those issues related to gender that affect the context and possibilities of women’s lives”) [36] should be considered as a soft skill and become an integral, and, over time, even natural, part of SEE.

It is possible to make modest but practical changes in SEE for improving SoWSE, as this paper has attempted to show. The proposed steps should come across as commonsensical to those teachers who understand SEE and are serious about contributing to making positive changes to SEE, and perhaps insightful to those students who are committed to improving the situation in which some of their peers continually find themselves in, and, in being so, are willing to make necessary changes to their own attitudes and/or behaviors, even if it comes at a personal cost. These steps are by no means exhaustive, and are only a beginning not the end, but then “a journey of a thousand miles begins with a single step”.

55
ACKNOWLEDGMENT

This paper is dedicated to all those women who, against all odds, continue to strive and thrive in software engineering education and profession today, hoping for a better tomorrow, and to all those men who, all the while enduring cost to self, unequivocally and unconditionally support these women in their pursuit. The authors are grateful to their women colleagues and students for sharing their experiences and to CUPFA for a Professional Development Grant.

REFERENCES

BDC: Using BERT and Deep Clustering to Improve Chinese Proper Noun Recognition

Yuanchi Ma  
Beijing Institute of Technology  
School of Computer Science and Technology  
beijing,China  
3220215164@bit.edu.cn

Hui He  
Beijing Institute of Technology  
Institute of Engineering Medicine  
beijing,China  
hehui617@bit.edu.cn

Zhendong Niu  
Beijing Institute of Technology  
beijing,China  
zniu@bit.edu.cn

Abstract—Proper noun recognition is a sub-task in named entity recognition. However, few methods have been specifically applied to the Chinese. The reason is that most of the existing deep clustering methods rely on manually labeled training sets, which take a long time in the learning process. And due to the wide and large-scale nature of the proprietary domain and the lack of word boundaries, recognizing Chinese specialized terms from unstructured text remains challenging. In this paper, we design an unsupervised method to improve Chinese proper noun recognition. The first step is to implement the word separation for Chinese, followed by a BERT-based improved word characterization method to obtain word vectors. Finally, we use the autoencoder-based deep clustering method to complete the extraction of proper nouns from books. We have done comparison experiments on the public dataset and our selected professional book data respectively, and the result is an improvement of our method in both the accuracy and F1 values. 1

Index Terms—Proper noun recognition, BERT, Deep clustering, GMM

I. INTRODUCTION

Proper noun recognition in NER is a key task that not only locates new terms in specialized fields but also identifies its entity classes from unstructured text, which can then be provided to various downstream NLP tasks for information acquisition. For example, question answering [1], relationship extraction [2], key information retrieval [3], [4], entity extraction and linking [5], hotspot discovery [6], etc. Compared with English, the task of extracting proper names for Chinese has been a major challenge due to the lack of word boundaries and large-scale manual annotation datasets. In detail, because of the unique language structure of the Chinese language, many word ambiguities occur [7], ignoring word-level information and using character-level information directly to recognize Chinese entities usually leads to poor performance, so Chinese word separation (CWS) needs to be performed to use word-level information to help determine word boundaries. For example, "Agricultural Bank of China Xingtai Branch" is a proper noun in the banking world. However, we can also mark "China", "Xingtai" and "bank" as separate entities.

Many types of methods nowadays need a large amount of labeled data for training neural networks [8, 9], but for data in unknown specialized fields, we do not have labeled data available for training. For the above problems, this paper proposes an unsupervised method to improve Chinese proper noun recognition (BDC). The first initial division of Chinese proper nouns is achieved by performing Chinese word separation. And then the word vector is obtained in the word vector representation module using the improved word characterization method based on BERT [10]. Finally, the word characterization results are imported into the autoencoder-based deep clustering network module, and the key features are extracted and mapped to the two-dimensional vector representation space to realize the word clustering of all unknown nouns in Chinese data, thus completing the clustering and extraction of proper nouns in Chinese. The process is shown in Figure 1.

In summary, we have made the following contributions to this paper:

- To the best of our knowledge, this is the first work to improve the recognition of proper nouns in Chinese based on deep clustering networks.
- We transform the NER task into an unsupervised clustering task, which can directly identify and extract proper nouns without a large number of labeled training data.
- We deal with complex book data and provide a unique feature vector representation space for proper nouns.
- We have carried out extensive experiments on three open Chinese NER data sets and selected Chinese book data, and the results show that our model achieves better results than the baseline NER model.

In the rest of this article, we will introduce the relevant work, the relevant methods, and the role of our model in the second section. In the third section, we introduce the implementation technology and process of our method in detail. In the fourth section, the experimental process and results are introduced, and ablation experiments are conducted to study the importance of each part of BDC. Finally, the paper is summarized in the fifth section.

II. RELATED WORK

The proper noun extraction tasks under the NER task follow the NER task classification and can be mainly classified into supervised and unsupervised approaches.
This flowchart shows each of the three important parts of this algorithmic network. The Participle Module is the first part, which segments the text part. The word vector characterization module is the second part, which implements word segmentation and converts it into word vectors. The deep Clustering Network Module is the third part, which is divided into two parts. They are autoencoder-based feature extraction (part1) and Identifying clusters of proper nouns (part2). Their roles are to extract word vector features and determine clusters of proper nouns after clustering, respectively. It should be noted that the adaptability of AE to Bert has been proven to be better after extensive experiments.

### A. Supervised and unsupervised methods

Among the supervised methods are dictionary-based methods, statistical machine learning-based methods, and deep learning-based methods. The unsupervised methods are mainly rule-based and clustering-based. Rule-based and dictionary-based methods were the first methods with named entity libraries and manual rules. Unsupervised based methods such as [11], which is an unsupervised method based on cyclic consistency training. Statistical machine learning-based methods mainly include Hidden Markov Models [12], Maximum Entropy Models (MEM), Support Vector Machines [13], and Conditional Random Fields [14], where CRF is the most widely used statistical NER method.

Most deep learning-based methods [15] inherit the classical lstm-crf or cnn-crf architecture. [16, 17] use convolutional neural networks to learn word representations from English NER characters. One of them [16] proposed a semi-supervised learning model based on the BiLSTM neural network with a large amount of unlabeled text and a rather limited amount of labeled text. [18] proposed a gated convolutional neural network model for Chinese NER. [19] proposed a novel word character LSTM model for Chinese NER, which adds word information to the beginning or end characters of a word.

### B. proper noun recognition

The above NER methods are very effective for generic NER vocabulary recognition, but not so effective for proper noun recognition in specialized fields. Recently, some professional domain proper noun recognition methods have been proposed, and they more or less borrow the ideas from the above methods. [20] A multilingual knowledge base based on Wikipedia is built to provide relevant contextual information for the Named Entity Recognition (NER) model to recognize proper nouns. To further improve the classification accuracy, word clustering information is added as feature embedding using K-means clustering method [21]. Language model equipped with access to an external knowledge base (KB). Our Knowledge-Augmented Language Model continues this work by augmenting traditional models with KBs, which fall into the category of unsupervised. In contrast, our approach takes full advantage of the best-performing word characterization methods coming in and gets a mapping of their features to a two-dimensional space based on autoencoder, which yields more comprehensive results to improve the performance of proper noun recognition.

### III. METHOD

#### A. Participle Module

We use jieba word division with the following principle: for existing words, a word graph scan is implemented based on a prefix lexicon to generate a directed acyclic graph (DAG) composed of all possible word formation cases of Chinese words in a sentence, and dynamic programming is used to find the maximum probability path to find the maximum cut-off combination based on word frequency. For unregistered words, an HMM model based on the word formation ability of Chinese characters is used, and the Viterbi algorithm is used for calculation.

#### B. Siamese Network model based on BERT-wwm

The BERT model [23] is one of the best-performing models in the field of word representation in recent years, and it can convert words into word vector representations, which is convenient for us to use in the next feature extraction work. [10] proposed a new improvement strategy based on the original BERT. We next used this model to achieve the characterization of all unknown nouns in the book data after using the overword for the text data extracted above. Their model proposes a new masking strategy, called MLM as a correction. In the original BERT, a word chunk tagger [10] is used to split the text into word chunk tags, where some words are split into several small fragments. Whole word masking mitigates the drawback of masking only a portion of the whole word, which is easier to predict for the model. In this model, we designed this part of the structure as the Siamese Network structure. As shown in Figure 1, part 1. That is, in training and testing, the sub-networks of the model use the BERT-wwm model, and the two models share parameters. The reason for this is to facilitate text standardization output and provide better vector space for the following in deep clustering.
C. Autoencoder-based feature extraction

For the obtained word vectors, we need to further obtain their key features. The role of the encoder [24] is to encode the high-dimensional input $x$ into a low-dimensional hidden variable $h$. This forces the neural network to learn the most informative features. The role of the decoder is to reduce the hidden variable $h$ in the hidden layer to its initial dimension. The best state is that the output of the decoder recovers the original input perfectly or approximately, i.e. $x' = x$.

Encoding process from the input layer to the hidden layer:

$$h = g_{θ_1}(x) = α(W_1x + b_1)$$ (1)

The decoding process from the hidden layer to the input layer:

$$\hat{x} = g_{θ_2}(h) = α(W_2x + b_2)$$ (2)

The optimization objective function of the algorithm is written as:

$$\text{MinimizeLoss} = \text{dist}(X, X^R)$$ (3)

where dist is the distance metric function of the two, which is usually calculated using the mean squared variance. If the number of neurons in the input layer $n$, is greater than the number of neurons in the hidden layer $m$, then we are equivalent to reducing the data from $n$ to $m$ dimensions. Then we use this $m$-dimensional feature vector to reconstruct the original data.

But to visualize the data, we still need to reduce the dimensionality. The dimensionality reduction should preserve the local features of the data. Then the data is visualized using the T-SNE [25] method for dimensionality reduction. The algorithm defines a nonlinear mapping from the feature space $M$ to the two-dimensional feature space $Z$. It minimizes the difference between the corresponding probability distributions of the mismatch $M$ and $Z$ in terms of pairwise distances by minimizing the asymmetry. Thus the feature space $Z$ obtained by these two consecutive steps is more suitable for estimating the number of clusters.

D. Identifying clustering modules

We use a GMM-based clustering algorithm [26]. There are four main criteria for judging the number of clusters using this method: the BIC value [27] of the model, the AIC value [28], the silhouette coefficient [29] and the calinski harabasz value [30]. The first step is to create a GMM model with a pre-set range of $k$ values for the number of clusters $k \subset [2, n]$, where $n = 3, 4, 5...$. After creating the set of models based on the range of $K$ values, the BIC values and AIC values under each model are calculated as a direct means of model evaluation. Where the AIC value describes the appropriate number of clusters and the BIC value assesses the simplicity and validity of the model. Calculation of the silhouette coefficients and CH values is used as an indirect means of evaluating the model. Our goal is to find the AIC and BIC values of the highest point of the silhouette coefficient, the CH value, and the lowest point at the relevant $k$-value. However, these values are fluctuating because of the existence of local optima. Therefore, from a statistical point of view, a Monte Carlo method is introduced to take expectations to avoid local optima and to derive the optimal number of clusters. In this way, we can perform GMM clustering analysis on the data based on the optimal number of clusters obtained. Among them, the probability density function of the GMM model is as follows.

$$P_M(x) = \sum_{i=1}^{K} p(k)p(x | k) = \sum_{i=1}^{K} \alpha_k p(x | \mu_k, \Sigma_k)$$ (4)

In this way, we can perform GMM clustering analysis on the data based on the optimal number of clusters obtained. In the GMM model of this paper, the aic value is calculated for each value of $k$. Let $n$ be the number of observations and $RSS$ be the residual sum of squares, then the AIC becomes as follows.

$$AIC = 2k + n\ln(RSS/n)$$ (5)

In BIC, $k$ is the number of model parameters, $n$ is the number of samples, and $L$ is the likelihood function. $k\ln(n)$ penalty term is used in cases where the number of dimensions is too large and

$$BIC = k\ln(n) - 2\ln(L)$$ (6)

In the silhouette coefficient, the intra-cluster dissimilarity is $a(i)$. The average of the dissimilarity of the $i$ vector to other points in the same cluster reflects the cohesiveness. The inter-cluster dissimilarity is $b(i)$. The minimum of the average dissimilarity of the $i$ vector to other clusters reflects the degree of separation.

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$ (7)

The CH values are calculated as follows. Where, $n$ denotes the number of clusters, $k$ denotes the current class, $trB(k)$ denotes the trace of the interclass departure matrix, and $trW(k)$ denotes the trace of the intra-class departure matrix.

$$CH(k) = \frac{trB(k)/(k-1)}{trW(k)/(n-k)}$$ (8)

At this point, we need to know the topic of the book has identified the clusters where the proper nouns exist. So next we perform a sub-word extraction and keyword extraction for the name of the book and the full text of the book respectively and select its extracted proper nouns and the first keyword as the subject words of the book. By mapping them to two-dimensional space by the above method, the category they belong to is the cluster of proper nouns we need to get at last.
IV. Experiments

A. datasets

Resume dataset. The dataset contains 4761 samples, 8 categories, for NAME, EDU, LOC, ORG, PRO, TITLE, CONT, and RACE.

CCKS2019 task1 dataset. Which has a total of 1379 samples, 6 categories, for anatomical sites, surgery, diseases and diagnoses, drugs, laboratory tests, and imaging tests.

WeiBo NER dataset. The dataset contains 1890 samples with 7 categories as LOC.NAM, LOC.NOM, PER.NAM, ORG.NOM, ORG.NAM, GPE.NAM and PER.NOM.

200 book dataset. The 200 e-books collected by this study whose fields include computer science, chemistry, etc. Teaching books from various Chinese universities in specialized fields contain a large number of proper nouns that are well suited for this study.

B. baseline

We compared our model on the three datasets mentioned above with the following open-source NER models. Since these baseline models were presented at different times, the different models apply to different datasets.

- [31]: Roberta-base-finetuned-cluener2020-Chinese
- [32]: Word2vec can express a word in vector form quickly and effectively through the optimized training model according to the given corpus
- [16]: It directly trains F1 scores instead of label accuracy, and trains F1 scores and label accuracy in a comprehensive way.
- [19]: It adds word information to the beginning or end character of a word and reduces the impact of word segmentation errors while obtaining word boundary information in the LSTM model of word character.
- [33]: The way to construct the graph neural network is to implement the Chinese NER as the graph node classification task through the dictionary.
- [34]: It proposes a simple and effective character-level Chinese NER representation method.
- [35]: This improves the performance of Chinese NER by integrating the structural information between Chinese characters.
- [36]: This is a method that combines Robert and CRF. Intended to help us explore detailed performance under different model structures.

C. Implementation details

In the experiments on the book dataset, to visualize the experimental results, we provide a demonstration of the clustering process of a book. As shown in Fig 2. Book is "MySQL Database Design and Application". The input layer is set to 768, and the hidden layer is set to 16 for the best results. The number of iterations is set to 100.

D. Experimental results

We tested three datasets on the baseline and BDC methods, respectively. Their accuracy, recall, and F1 values are calculated. From the experimental results in Tables 1 and 2, our method obtained the best F1 value on the CCKS2019 and books data sets respectively, especially on the book data set. The other two data sets have better baseline results. The reason we analyzed is that Resume and WEIBO NER data sets are more general and less professional, while CCKS2019 and books data sets in the computer field have more professional terms. They focus on the professional terms of the medical field and computer field respectively.

<table>
<thead>
<tr>
<th>baseline</th>
<th>Resume P(%)</th>
<th>Resume R(%)</th>
<th>Resume F1(%)</th>
<th>CCKS2019 task1 P(%)</th>
<th>CCKS2019 task1 R(%)</th>
<th>CCKS2019 task1 F1(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31]</td>
<td>93.02%</td>
<td>92.74%</td>
<td>92.88%</td>
<td>11.11%</td>
<td>58.85%</td>
<td>18.69%</td>
</tr>
<tr>
<td>word2vec[32]</td>
<td>-</td>
<td>-</td>
<td>85.89%</td>
<td>8.89%</td>
<td>80.00%</td>
<td>14.55%</td>
</tr>
<tr>
<td>WC-LSTM[19]</td>
<td>95.21%</td>
<td>95.10%</td>
<td>95.15%</td>
<td>51%</td>
<td>53%</td>
<td>55%</td>
</tr>
<tr>
<td>LGN[33]</td>
<td>95.20%</td>
<td>95.44%</td>
<td>95.36%</td>
<td>45%</td>
<td>43%</td>
<td>39%</td>
</tr>
<tr>
<td>LSTM[34]</td>
<td>95.44%</td>
<td>95.76%</td>
<td>95.60%</td>
<td>73%</td>
<td>71%</td>
<td>62%</td>
</tr>
<tr>
<td>MECT[35]</td>
<td>96.45%</td>
<td>95.38%</td>
<td>95.91%</td>
<td>36%</td>
<td>56%</td>
<td>41%</td>
</tr>
<tr>
<td>Roberta+CRF[36]</td>
<td>94.50%</td>
<td>95.00%</td>
<td>94.75%</td>
<td>31.32%</td>
<td>60.77%</td>
<td>41.34%</td>
</tr>
<tr>
<td>BDC</td>
<td>93.57%</td>
<td>84.72%</td>
<td>88.93%</td>
<td>86.33%</td>
<td>56.19%</td>
<td>68.07%</td>
</tr>
</tbody>
</table>

E. Ablation experiments

In the ablation experiment, we want to explore the effect of different modules on the overall network structure and results. In this algorithm, there are three modules. Among them, the...
participle module is a pre-processing of the book data, so we use the word vector representation module and the deep clustering network module as variables to explore which part has more influence on this algorithm.

We replace the word vector representation module and the deep clustering network module of the algorithm with other methods, respectively. The word vector characterization module is replaced by word2vec as the word characterization method, which we call BDC-2vec. The deep clustering network module uses k-means instead of GMM clustering, and instead of using the method of automatically determining the number of clusters based on parameters, only the contour coefficients are used as the criterion for determining the number of clusters. We call it BDC-kmeans.

According to Table 3, we can see that BDC-2vec works better than BDC-kmeans and the original BDC works last. Thus for the overall algorithm, the deep clustering network module has more impact and this is where we innovate the most.

V. CONCLUSION

In this paper, we propose a large-scale network architecture for the Chinese language to extract the technical terms among them. We combine a pre-training model based on BERT improvement with an autoencoder based deep clustering model to achieve the transition from a proper noun extraction task to a clustering task. We design multiple working modules to extract key features from the learned information for proper noun prediction. Extensive experiments show that our proposed method reaches at least the state-of-the-art baseline. The effectiveness in a good representation of this unsupervised proper noun extraction method. In addition, the word vector representation of the pre-trained model is slow due to the large vocabulary of many books’ data. In the future, we will investigate how to reduce the model parameters and improve the training speed while enhancing the experimental results.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 62272048.

TABLE II: This table shows the scores of BDC on two hundred dedicated book datasets. We calculated Precision, Recall, and F1 values separately.

<table>
<thead>
<tr>
<th>baseline</th>
<th>WEIBO NER</th>
<th>Book dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P(%)</td>
<td>R(%)</td>
</tr>
<tr>
<td>[31]</td>
<td>72.38%</td>
<td>41.57%</td>
</tr>
<tr>
<td>[16]</td>
<td>70.12%</td>
<td>40.66%</td>
</tr>
<tr>
<td>WC-LSTM[19]</td>
<td>66.10%</td>
<td>31.98%</td>
</tr>
<tr>
<td>LGN[33]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LSTM[34]</td>
<td>71.86%</td>
<td>49.11%</td>
</tr>
<tr>
<td>MECT[35]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Roberta+CRF[36]</td>
<td>70.44%</td>
<td>34.67%</td>
</tr>
<tr>
<td>BDC</td>
<td>72.17%</td>
<td>36.83%</td>
</tr>
</tbody>
</table>

REFERENCES

<table>
<thead>
<tr>
<th>BDC-2vec</th>
<th>BDC-kmeans</th>
<th>People’s Daily 2014</th>
<th>200 books</th>
<th>800 books</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>P(%)</td>
<td>R(%)</td>
<td>F1(%)</td>
</tr>
<tr>
<td>✗</td>
<td>✓</td>
<td>95.15%</td>
<td>95.21%</td>
<td>51%</td>
</tr>
<tr>
<td>✓</td>
<td>✗</td>
<td>95.39%</td>
<td>95.98%</td>
<td>36%</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>73.3%</td>
<td>41.57%</td>
<td>60.32%</td>
</tr>
</tbody>
</table>


Session SEKE2: Software Engineering and Knowledge Engineering
Applying Reinforcement Learning for Automated Testing of Mobile Application
Focusing on State Definition, Reward, and Learning Method

Keita Murase
Dept. of Info. and Comp. Sci., Keio University
murase42@doi.ics.keio.ac.jp

Shingo Takada
Dept. of Info. and Comp. Sci., Keio University
michigan@ics.keio.ac.jp

Abstract

There have been various studies on the automation of mobile app testing. Typical methods for automated testing of mobile apps are based on random search and on building state transition models. But there are problems in terms of the efficiency of search and accuracy of model building. This paper focuses on applying reinforcement learning to testing of mobile apps, especially issues such as explosion of the number of states, fixed rewards for transitions, and difficulty in convergence of learning. We focus on state definition, reward function, and a learning method to solve these problems. Specifically, we define states using discrete values of UI (User Interface) information on the screen, define a dynamic reward function, and perform periodic learning by using the transition history. The proposed method is implemented and evaluated. Evaluation results show that our proposed approach shows 1.21 times higher coverage than an existing tool using reinforcement learning.

1. INTRODUCTION

Mobile apps are applications that run on smartphones and other devices, and play a very important role in our daily life. Among smartphone OSs, Android has the largest market share. Android apps need to be tested just like any other software, such as through GUI testing. GUI testing of Android apps involves actually performing operations on the screen displayed on the device and checking for anomalies. However, manual GUI testing is very expensive when the scale of the application is large or the frequency of updates is high. Automated GUI testing will reduce this cost.

Several methods have been proposed for Android GUI testing. One of the most well-known methods of GUI test automation is random testing. Random testing randomly selects and executes operations on the screen, and is used in the well-known tool Monkey [9]. However, since the selection of operations is random, there are problems with the efficiency and stability of the search. For improving the search efficiency, model-based methods were proposed [12][4]. In the model-based method, the available actions and the states that can be reached by the actions in each state are obtained in advance by static analysis of the code and are represented in the form of a state transition graph enabling efficient path searches. However, accurate analysis of Android apps is difficult, and it is often impossible to construct state transition diagrams correctly.

Reinforcement learning is a well-known method for automatic game playing, and recently, several methods have been proposed using reinforcement learning for automatic testing of Android [6][7][8]. In reinforcement learning, rewards are given for state transitions, and search strategies are learned so that the rewards obtained are large. The advantage of this method is that it does not depend on the accuracy of the model since the strategy is based on the actual state transitions.

However, most approaches give the same reward to the same transition regardless of the situation, which may lead to the same transition to be always chosen. Thus, we propose a method to improve the coverage of the test by giving rewards for reinforcement learning according to the search situation. Naive application of this may lead to difficulty for the learning results to converge. Thus, we propose a method to cope with this convergence problem by storing the history of transitions, and periodically repeating the learning focusing on the most recent transitions.

This paper is organized as follows: Section 2 provides a brief introduction to reinforcement learning. Section 3 reviews related work. Section 4 describes our proposed approach and implementation. Section 5 evaluates our approach and section 6 makes concluding remarks.

2. REINFORCEMENT LEARNING

Reinforcement learning proceeds through the interaction between the environment and the agent. The basic flow is as follows:
1. The agent performs an action based on a strategy.
2. Based on the agent’s action, the environment changes its state.
3. The environment rewards the agent as a result of the action.
4. The agent improves its strategy based on the reward.

As we will show in the next section, reinforcement learning has been applied to GUI testing of Android apps, where environment is the Android device, an action is an operation on the Android device, and state is a state of the screen.

Q-learning [10] is a popular reinforcement learning algorithm. It represents a strategy in terms of a Q-function, and is an algorithm for learning a Q-function. The Q-function $Q(s, a)$ is the value of the action $a$ in the state $s$. If this value is correct, the reward can be increased by choosing an action with a large Q-function in each state. If an action $a_t$ in state $s_t$ results in a transition to state $s_{t+1}$ and a reward $r_t$, the value of $Q(s_t, a_t)$ is estimated by Q-learning as in Formula (1). $\gamma (0 \leq \gamma \leq 1)$ is the discount rate, which indicates how much of the future reward is taken into account.

$$Q(s_t, a_t) = r_t + \gamma \max Q(s_{t+1}, a_{t+1}) \quad (1)$$

With the learning rate $\alpha (0 \leq \alpha \leq 1)$, which indicates how much the Q-function is changed, the value of the Q function can be updated as in Formula (2).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha (r_t + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (2)$$

3. RELATED WORKS

Monkey [9] is one of the most widely used tools for automated testing of Android apps. The tool randomly generates events such as click operations. However, since the selection of actions is random, there are problems in terms of efficiency and stability of the search. In this study, we aim to improve the efficiency of search by using reinforcement learning.

Some studies have attempted to construct a state transition model and search on it. Yang et al. [12] proposed Window Transition Graph, which supports multiple windows and system events, and performed path exploration and test case generation on the graph. Lai [4] et al. proposed Screen Transition Graph, which represents the state transitions of the Android screen by static analysis, considering background execution and screen composition. They have also developed Goal Explorer, which automatically and efficiently tests specific functions by prioritizing the direction near a specific function on the graph. The problems with these methods are that they are prone to omissions of states and transitions when analyzing and modeling Android apps and that they are unable to deal with cases in which transitions include randomness. When reinforcement learning is used, efficient search can be performed by taking probabilities into account based on the actual execution results.

There are also approaches that focus on heuristics. TimeMachine [2] saves various states that are expected to be visited as “interesting states” and takes a snapshot of the machine state so that it can be resumed from that state at any time. Then, when the machine is in a situation where it only visits states that it has already visited, it can resume from the interesting state to facilitate the search.

Recent testing work has applied reinforcement learning for testing Android apps [8][7][6]. Qdroid [8] is based on Deep Q-Network. It groups GUI components according to their semantics and decides which group of GUI components to act on. The reward is larger when the screen changes, making it easier to avoid meaningless actions that have no effect on the screen. However, if the reward does not change with time, the actions taken in each state will become constant when the learning process is completed. This results in the Q-function to converge to a constant value, causing the same transition to be repeated. This is contrary to the purpose of the test, which is to perform a wide variety of transitions. ARES [7], like Qdroid, gives a large reward when a screen change occurs, and also gives a reward when a bug is found. Similar to Qdroid, it is highly likely that the same transitions that eventually yield high rewards are repeated after the learning process converges. Q-testing [6], on the other hand, introduces rewards that change with time. It uses Siamese Network to judge whether the screen is similar to a screen already visited, and gives higher rewards when a similar screen has not been visited. However, when the reward changes over time, the problem arises that learning may not converge.

4. PROPOSED APPROACH: IMPROVING REINFORCEMENT LEARNING ALGORITHM FOR EXPLORATION

Based on the problems in the previous studies, we focus on three aspects of applying reinforcement learning, and propose a definition for each of them. First, we focus on state definitions, and characterize the screen in such a way that the number of states is reduced by using information from UIAutomator [11]. Second, we focus on the definition of rewards, and define it so that it changes according to the current search status. Finally, we focus on the learning process, and make it iterative so that the reward converges even if the reward is dynamic. We describe each of these in more detail in the rest of this section.
4.1. State and Action

The state definition is based on the following attributes, which are obtained from UIAutomator [11].

- Resource ID
- Possible operations: clickable, long clickable, scrollable, checkable, focusable

Resource ID is a feature assigned to each UI element. For example, Figure 1 is the initial screen of the card game Hot Death [3]. The NEW GAME button has the Resource ID com.smorgasbork.hotdeath:idbtn_new_game. In the screen shown in Figure 1, all five buttons are assigned different Resource IDs, but there are cases where multiple UI elements have the same Resource ID. This will be handled using actions, described later in this subsection.

UIAutomator also provides information about possible operations, such as whether each UI element is clickable or not, and whether it is long-clickable or not.

![Figure 1. Screen 1](hotdeath.png) ![Figure 2. Screen 2](hotdeath.png)

Based on information obtained by UIAutomator, we group the items whose Resource IDs and possible operations are the same, and define the state using the size of each group. For example, in Figure 1, if we assign five buttons to the first five components, the state is defined as the vector \((0, 1, 1, 1, 1, 1)\). Note the “0” in the first element. UI elements that are not currently on the screen are also included in the state definition where the corresponding element size is 0. Thus, Figure 1 has an element that is not visible. In Figure 2, we can now see the CONTINUE button, which makes the vector to be \((1, 1, 1, 1, 1, 1)\). The size that is necessary for a screen depends on the app, but Hot Death requires about 100-dimensional vectors. Coordinates of each UI element can also be obtained with UIAutomator, but since they have continuous values, we have omitted them to avoid an explosion of the number of states.

For a given UI element, one of the possible operations is chosen as an action. At this point, an action value is defined for the pair of state and that UI element. There may be a case where there are multiple UI elements with the same Resource ID. Although we do not save the coordinates of the UI elements, the order of the UI elements is based on the coordinates. This enables us to calculate the action value even if there are multiple UI elements with the same Resource ID. Note that we can also check if the action is a valid operation for the the corresponding UI element. For example, a click is always applied to a button for which only the clickable attribute is true, and a random string is an input to a UI element of the EditText class.

4.2. Reward

First, we define the penalty \(\text{penalty}(s_t, a_t)\) for performing action \(a_t\) in state \(s_t\) as in the following Formula (3).

\[
\text{penalty}(s_t, a_t) = \sum_s \frac{\text{count}(s, a_t)}{\text{distance}(s, s) + 1}
\]  

(3)

where \(\text{count}(s, a_t)\) is the number of times an action \(a_t\) has been performed in state \(s\) in the past, and \(\text{distance}(s_t, s)\) is the Manhattan distance between the vectors of state \(s_t\) and state \(s\). In other words, the more times the same action has been performed and the more similar \(s_t\) is to the state in which the action was taken, the larger penalty is given.

With the penalty, we define the reward \(r(s_t, a_t)\) for action \(a_t\) in state \(s_t\) as in the following Formula (4).

\[
r(s_t, a_t) = \begin{cases} X_{\text{high}} & (\text{penalty}(s_t, a_t) < P) \\ \frac{X_{\text{high}}}{\text{penalty}(s_t, a_t)} & (\text{penalty}(s_t, a_t) \geq P) \end{cases}
\]  

(4)

If the penalty is less than the threshold \(P\), a large constant reward \(X_{\text{high}}\) is given, and if the penalty is above the threshold, the small reward reduced according to the penalty is given. The reason for the division according to the threshold is to emphasize the importance of the first transition which has a large significance in the test. In the above reward definition, the reward changes according to the situation of the search. It leads to selecting actions according to the situation.

In Figure 1, consider the penalty \(\text{penalty}(s_t, a_t)\) for selecting the HELP button. Since the distance from the state of Figure 1 is naturally 0, if the HELP button has been selected three times in the past, \(\frac{3}{4}\) is added to the penalty. Since the distance from the state of Figure 2 is \(\text{distance}((0, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1, 1)) = 1\), if the HELP button has been selected three times in the past in Figure 2, the penalty is \(\frac{3}{5}\). If the HELP button is selected three times in each of the two screens, the penalty is 4.5, and the reward is \(\frac{X_{\text{high}}}{4.5}\), if this is greater than or equal to the threshold value \(P\).
4.3. Learning

Q-function can be kept in a tabular form with each row as a state and each column as an action. The value of each cell represents the value of an action in a certain state. When a new state is visited, a new row is added and the reward is initialized based on Formula (4). Since it is difficult for the Q-function to converge by simply learning based on the update formula of the Q-learning at each state transition, the learning is periodically iterated as follows:

1. Recalculate the reward for the previous transitions.
2. Take the last $N$ transitions in the order of newest to oldest and repeat updating the Q-function.
3. Randomly select the previous transitions, and repeat updating the Q-function.

First, we recalculate the transitions that have been executed, since it is highly likely that the rewards have changed since the transition occurred. Next, we iteratively train with the most recent transitions that may not have been reflected yet. In this iteration, the transitions are taken from the most recent ones so that the most recent results can be efficiently propagated to the traversed routes. Finally, we aim to bring the overall Q-function close to the correct value by repeating the learning process with a random selection of all the transitions up to now.

4.4. Implementation

We implemented our tool based on Qdroid [8] (Figure 3). Interactor interacts with the Android app by using UIAutomator [11]. Interactor separates the screen data in XML format received from UIAutomator into an array of UI elements and sends the UI information to the Agent. Menu and Back buttons are added as virtual elements to the array even if they do not appear on the screen, so that they can be handled if they are operable. We also add randomly positioned buttons to the array. These buttons are added as UIAutomator may not recognize some UI elements.

The Agent obtains UI information from Interactor, selects the next action, and updates the Q-table. When selecting the next action, Agent selects the action which has the largest Q-function with high probability, and selects the action randomly with low probability, based on $\epsilon$-Greedy. The selected action information is sent to Interactor, which then performs that action through UIAutomator. Additionally, Interactor collects stack trace of crashes with logcat [5].

5. EVALUATION

We consider the following three research questions for evaluation:

- RQ1: How is the performance compared to other tools?
- RQ2: How much influence do our changes have?
- RQ3: How does coverage change over time?

Evaluation was performed on the virtual environment provided by Androtest [1] in the following execution environment:

- OS: Ubuntu 14.04.1 LTS
- CPU: AMD Ryzen Threadripper 3990X 64-Core
- Memory: 6113MB
- Emulator: Android 4.4

The target apps are the same as Qdroid. We measured the average method coverage and the average number of unique crashes over three trials for each app.

5.1. RQ1: How is the performance compared to other tools?

We targeted Qdroid, ARES and Q-testing for our evaluation. Unfortunately, although the Q-testing executable was made available$^1$, we could not make it execute in our environment. A similar situation occurred with ARES$^2$, where we could make it execute for some apps, but not the ones we were targeting. Thus, we focus on Qdroid for comparison.

Table 1 shows the results of the coverage and the number of crashes between Qdroid and our proposed approach (“Our tool”). We will discuss Qdroid2 in RQ2.

Both coverage and number of crashes showed the same trend. First, the coverages of Anymemo, Multi SMS Sender, MunchLife, and Weight Chart were improved. This is considered to be due to the increase of search space by the changes in the UI recognition method, including the addition of virtual UI elements. For example, in MunchLife, menu button

$^1$https://github.com/anlalalu/Q-testing
$^2$https://github.com/H2SO4T/ARES
Table 1. Evaluation results

<table>
<thead>
<tr>
<th>Application</th>
<th>Qdroid</th>
<th>Qdroid2</th>
<th>Our tool</th>
<th>Qdroid</th>
<th>Our tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any Memo</td>
<td>43.2</td>
<td>47.1</td>
<td>49.3</td>
<td>3.7</td>
<td>6.0</td>
</tr>
<tr>
<td>Dalvik Explorer</td>
<td>82.6</td>
<td>80.5</td>
<td>83.7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hot Death</td>
<td>64.8</td>
<td>79.0</td>
<td>80.2</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>Mileage</td>
<td>38.2</td>
<td>43.2</td>
<td>50.4</td>
<td>0.7</td>
<td>1.7</td>
</tr>
<tr>
<td>Mini Note Viewer</td>
<td>59.6</td>
<td>51.4</td>
<td>48.7</td>
<td>1</td>
<td>0.7</td>
</tr>
<tr>
<td>Multi SMS Sender</td>
<td>37.9</td>
<td>64.7</td>
<td>66.9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Munch Life</td>
<td>53.8</td>
<td>92.3</td>
<td>92.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>My Expenses</td>
<td>62.5</td>
<td>69.8</td>
<td>82.7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Random Music Player</td>
<td>58.7</td>
<td>58.7</td>
<td>58.7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tipp Tipper</td>
<td>56.1</td>
<td>82.8</td>
<td>88.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Weight Chart</td>
<td>46.1</td>
<td>64.2</td>
<td>73.6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Who has my stuff</td>
<td>89.1</td>
<td>75.6</td>
<td>83.7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Average</td>
<td>57.7</td>
<td>66.7</td>
<td>69.8</td>
<td>5.4</td>
<td>9.1</td>
</tr>
</tbody>
</table>

5.2. RQ2: How much influence do our changes have?

There are two major differences between our approach and Qdroid: the method of acquiring UI information (adding virtual UI elements) and the reinforcement learning algorithm (state definition, reward function, and periodic learning). In order to measure the influence of the change in the reinforcement learning algorithm, we implemented the part of Qdroid related to the UI acquisition and the operation of the app in the same way as the proposed method and measured the coverage. We call this tool “Qdroid2”. The results are shown in Table 1.

The results show that the change of the UI acquisition method (i.e., Qdroid2) improved the coverage of Qdroid from 57.7% to 66.7%, and the change of the reinforcement learning algorithm further improved the coverage to 69.8%. We believe that our UI acquisition method led to a widening of the search area leading to the increase in coverage.

However, some apps showed a decrease in coverage. For example, the coverage dropped from Qdroid to Qdroid2 in Dalvik Explorer, Mini Note Viewer, My Expenses, and Who has my stuff. This suggests that the addition of virtual UI elements will lead to an increase in the number of extra action options, which may be unnecessary. Still, except for Mini Note Viewer, the coverage for our tool increased from Qdroid2 to at least nearly the same result as Qdroid due to changes in the reinforcement learning algorithm, most likely due to avoiding unnecessary actions.

5.3. RQ3: What is the change of coverage over time?

Figures 4 and 5 show how coverage changed over a two hour period for Munch Life and Mileage, respectively. The coverage was measured every five minutes, and the results are the average of three trials. Including other apps which are not shown (due to space), most of the apps show a rapid increase in coverage immediately after starting, followed by a gradual increase in coverage. This makes sense as many codes are involved in the process immediately after starting.

The coverage in some apps, such as Munch Life (Figure 4), reached its maximum value at an early stage. This is likely because the app was small and thus the searchable portions were quickly exhausted. Another app that showed this same trend was Random Music Player, where a particular type of input, in this case appropriate URLs, were necessary. Our tool cannot automatically generate such URLs, but this is an area of future work.

The coverage in other apps, such as Mileage (Figure 5), was still increasing after two hours, especially for our tool. These apps were large or require complicated procedures to execute their functions. For example, Mileage has a wide variety of settings for functions related to the mileage of a car, such as the type of car, the unit of distance, etc. Any Memo and My Expenses were two other examples of large app showing an increase after two hours. On the other hand, Hot Death also was fairly large, but it did not increase as much, most likely due to code related to winning which is difficult to reach with automatic execution.

5.4. Threats to Validity

One threat to validity is the number of apps tested which was 12. We chose these 12 as they were used in other research. But since even our results showed that coverage varies greatly depending on the app, we consider further evaluation as future work.
Second, we were only able to compare our tool with Qdroid. We need to compare our approach with other tools.

Third, in the experiments, we conducted three 2-hour trials for each app. However, there were some apps for which the coverage did not change at all after 2 hours. Since the execution time determines whether the speed of the search or the size of the search space is more important, it is possible that the results will change significantly by changing the execution time. In addition, there are apps where the coverage is almost the same each time, while there are also apps where the coverage fluctuated by more than 10%, suggesting that the number of trials may not be sufficient for some apps.

6. CONCLUSION

We applied reinforcement learning to test Android apps. Other researchers have done work on this, but we especially focused on (1) state definition so that the number of states does not explode, (2) a reward that changes depending on the situation, and (3) a learning approach to make full use of them. As a result, we were able to improve the coverage compared to Qdroid, which our implementation was based on.

Future work includes further evaluation and improvement of our proposed approach. As for the evaluation, more apps should be included in the experiment to make the evaluation more generalizable. We also expect that the performance of our approach can be improved by extending the supported UI actions and adjusting various parameters, states, and rewards.

ACKNOWLEDGEMENT

This work was supported by JSPS KAKENHI 22K11979.

References

A Model-Driven Development Framework for Geographical and Relational Databases Systems

Carlos Eduardo Pantoja  CEFET/RJ  Universidade Federal Fluminense  pantoja@cefet-rj.br

Tielle da Silva Alexandre and José Viterbo  Universidade Federal Fluminense  tiellesa@id.uff.br, viterbo@ic.uff.br

João Victor Guinelli  Petrobrás  CEFET/RJ  joao.silva@cefet-rj.br

Abstract—The database project is the process of engineering a database model from conceptual modeling to database implementation. Existing tools allow the conceptual modeling of relational and geographical databases separately, but none integrates both in a single solution. This paper presents an Model-Driven Development framework for creating relational and geographical database models. The framework comprises an extended relational metamodel to adhere to geographical database concepts present in the OMT-G, an Entity-Relationship modeling tool, Query View Transformation rules between OMT-G and the extended metamodel, and Model-To-Text transformations to generate ANSI SQL/SFS code.

Index Terms—MDA, database, software engineering

I. INTRODUCTION

The database project consists of different abstraction levels that converge to the database implementation using a Database Management System (DBSM) and a structured language. A database system project can be considered a set of sequenced transformations from high abstraction models to a specific platform model. At the early level, the database requirements gather textual business information necessary to create a database structure. Afterward, the database designers, considering a modeling technique, transform this textual business description into a graphical model. The conceptual modeling is responsible for describing the business information model using a graphical notation at a higher abstraction level. It allows an early visualization of the structure of a database and how these structures are related to each other [1].

The next level results from a transformation from conceptual database modeling to a logical model. The logical model considers additional constructions to the conceptual database modeling to avoid anomalies, redundancy, and inconsistency. This level does not consider any implementation characteristic using a specific technology or language. However, it can consider the kind of database modeled, e.g., relational databases. The physical model is the consequence of transforming from the logical model considering a target DBSM and a database language. This level considers some specific technologies for the implementation of the modeled database.

There are several modeling languages and notations for modeling a relational database, such as ER [2]; Crow’s Foot [3]; and IDE1FX [4]. Several modeling options exist for a geographical database system, such as UML-GeoFrame [5], [6] and the OMT-G [7]. However, these languages do not offer a solution integrating different models and approaches. In general, a geographical database system cannot be projected using a relational notation; otherwise, it is possible to model a relational system using some constructions present in geographical notation since it extends the relational concepts.

Conversely, Model-Driven Engineering (MDE) provides a software development process based on model transformations that separate the business modeling from specific technologies until effective software implementation. The Model-Driven Architecture (MDA) specifies abstraction levels from technology-independent models to a specific platform model to assist software development. The MDA uses a Computational Independent Model (CIM); a Platform Independent Model (PIM); and a Platform Specific Model (PSM) [8].

Applying the MDA in the software development process implies using metamodels, which define the concepts of another model at a lower level. Several metamodels intend to provide concepts for a database system, such as the Common Warehouse Metamodel (CWM) [9] for interchanging information between the Meta Object Facility (MOF) [10], Unified Modeling Language (UML) [11] and XML Metadata Interchange (XMI). It provides a generic model for several languages and notations, including the relational model. However, it does not include any geographical concept. It provides a simple mapping between and oriented-object concepts, the Enhanced Entity-Relationship (EER) [12] metamodel, which provides concepts for all ER constructions using Chen’s notation. Although a case tool supports the metamodel, it does not provide constructions for other modeling languages, such as Crow’s Foot; and the Generic Database Modeling Metamodel (GEDBM) [13], which gathers concepts of relational database modeling languages and several notations. The metamodel is supported by a semi-automatic tool and a SQL code generator but is strictly directed toward relational concepts.

This paper presents a framework for designing database systems, which uses a metamodel for relational and geographical database system projects that gathers the concepts of several modeling languages and notations. The relational part was constructed by observing the extent modeling languages and notations used in the database development process. The geographical part of the metamodel was constructed by ob-
serving the concepts of the OMT-G model. The methodology is composed of the revised GEDBM, which can be instantiated by a set of Query-View-Transformation (QVT) [14] rules from an adherent modeling language and; an existing CASE tool or by a new one constructed from the GEDBM concepts. A set of QVT rules from the OMT-G model to the GEDBM will be defined to prove the validity of geographical databases.

After that, a set of Model-To-Text (M2T) [15] rules for ANSI/SQL and SFS/SQL specifications can be generated from the GEDBM independently of the modeling language used; the GEDBM will be constructed using the Ecore from Eclipse Modeling Framework (EMF) [16]; the QVT rules will be implemented using EMF’s Model-To-Model (M2M); the M2T rules will be implemented using Accceleo [17]; and the graphical ER tool will be constructed using the Graphical Modeling Framework (GMF). We aim to provide a framework for both database teaching and development. Besides, it can be extensible for several notations and modeling languages, amplifying the framework’s applicability in various domains. We also provide some set of transformations between different models and target platforms to facilitate the database development process (for both relational and geographical databases).

II. Metamodels for Designing Database Systems

The GEDBM metamodel is based on the conceptual and logical database project and is generic for the most used relational modeling languages and notations. The modeling languages adherent to GEDBM is ER Chen’s Notation, Crow’s Foot, and IDE1FX. The metamodel consists of meta-classes such as Database, Relationship, Entity, and Field that represent aspects of conceptual modeling languages. Besides, it defines others like PrimaryKey, ForeignKey, and Check that represent aspects of the logical project, such as the uniqueness of a record, referential integrity, and attributes integrity. Thus, it is possible to build graphical tools for relational modeling languages using this metamodel. In the same way, if any tool exists with a modeling language adherent to GEDBM, it can be integrated into the GEDBM core by mapping its concepts.

The OMT-G metamodel is used to model geographical databases. The meta-class Schema contains the meta-class Element and the meta-class BaseRelationship that represent entities and relationships, respectively. The Element is specialized in Conventional, which can be geoObject and geoField. Thus, the created entities can represent three data types that exist in the OMT-G model: non-spatial (Conventional), discrete (geoObject), and continuous (geoField).

Furthermore, in the OMT-G, there are five different meta-class types to represent geographic fields (Network Class, Isoline, Sampling, Tessellation, and AdjacentPolygons), so the class geoField is specialized in five meta-classes that represent these types. The same thing occurs with the class geoObject, this meta-class is specialized in geoObjectWithGeometry and geoObjectWithGeometryAndTopology, and these two meta-classes are specialized too: the first one in Polygon, Point, and Line; and the second one in Node, UnidirectionalLine, and BidirectionalLine. Similarly, the meta-class baseRelationship, which represents the relationship between two entities, is specialized in association, Aggregation, generalization, SpatialAggregation, and cartographicGeneralization.

III. Methodology

In this section, we present the MDD methodology (Figure 1) for relational or geographical database projects. In the relational part we use the GEDBM metamodel, which gathers generic concepts for several modeling languages. Our extended metamodel uses the OMT-G to ensure the representation of geographical concepts. Furthermore, we present a set of M2M transformation rules for mapping concepts from the OMT-G to the GEDBM metamodel and a set of M2T transformation rules, which transfers the instance of the GEDBM metamodel in SQL and SFS code. The proposed methodology relies on the MDA to specify metamodels and transformation sets.

![Fig. 1. The proposed MDD approach.](image)

The MDA is a software development approach created by OMG where the metamodel has an important role in software development. In the MDA, the transformation process among models in different levels of abstraction is responsible for the software construction. The MDA approach contributes to the development of applications more independent of technology once models can be combined and new models from new technologies can be later added. It increases the interoperability of the project and facilitates its maintenance.

Our methodology starts by selecting one of the several modeling languages that can be used in GEDBM or the OMT-G model. In this step, it is possible to interconnect existing tools to instantiate these models once their concepts adhere to the ones present in GEDBM. Suppose it is chosen the ER modeling tool. In that case, the GEDBM is directly instantiated (in this case, the modeling tool should be constructed using the GEDBM, or its metamodel should adhere to the GEDBM). If the OMT-G model is chosen, it is possible to execute a transformation using the QVT language between the chosen model and the GEDBM metamodel. Otherwise, the GEDBM can be manually instantiated. The Object Constraint Language (OCL) is used in the metamodel validation to guarantee correctness (with minimum concepts required for GEDBM).

After that, it is possible to generate ANSI SQL 92/99/03 or SFS/SQL standard code using a set of transformation rules that uses the M2T specification. The SFS is a specification proposed by OGC that defines how spatial or vector components of geographical data should be stored in a database; and
how to store, read, query, and update these components using SQL. The PostgreSQL and Oracle Database are widely used DBMS with geographical extensions based on the SFS/SQL.

Besides, other tools, models, and target platforms can be added to the methodology. When it is desired to integrate an existing tool, it must use the GEDBM or the OMT-G as metamodel; or its metamodel must comply with the GEDBM. In this case, it is possible to specify a new QVT transformation from the existing model to the GEDBM. Finally, another target platform can be added by creating a new set of transformation rules in M2T from GEDBM to the target platform.

A. The Revised GEDBM Metamodel

The methodology uses the GEDBM as the core of the MDA approach for database design. However, the original GEDBM is not prepared to support concepts from geographical modeling languages and models. Therefore, we present a modified version of the GEDBM metamodel, where some modifications were performed to add geographical concepts to the GEDBM. The modifications include analyzing if geographical constructions from the OMT-G model adhere to the GEDBM. The metamodel also accepts the min-max notation. The GEDBM metamodel can be seen in Figure 2.

In the OMT-G, entities have different types. So, it was necessary to extend the representation of entities and relationships so that they could represent geographical types that do not exist in relational modeling. For this, it was added the enumerations EntityType and RelationshipType. So, as primary keys and foreign keys are not represented in some modeling languages in the conceptual model, we decided to substitute them. The meta-class Field was modified to a common field (CommonField) or an identifier field (IdentifierField), and they can refer to an Entity. Now, the rules to generate SQL code are simpler: the foreign keys are automatically generated, and the primary keys result from instances of the IdentifierField.

Additionally, the concepts of the weak and associative entity were added to the metamodel. Then, the meta-class Entity was modified to be the parent of common entities (CommonEntity) and associative entities (AssociativeEntity). Furthermore, a common entity can now be represented as a StrongEntity or WeakEntity. So, the metamodel is now formed by a base metaclass DataBase that contains Entities and Relationships with two or more Cardinalities. There is an associated cardinality for each entity, so it is possible to represent the relationships between an entity and itself and between two or more entities. Moreover, an AssociativeEntity refers to exactly two other entities without a relationship between them (the associative entity is a result of the relationship of these two entities). Besides, an Entity has many Fields that are specialized in CommonFields or IdentifierFields.

B. From OMT-G Model To GEDBM Model

The first set of transformation rules — QVT — uses the OMT-G as the source Platform Independent Model and the GEDBM as the target Platform Independent Model. This transformation set will generate an equivalent GEDBM model with the occurrences of geographical concepts presented in the OMT-G modeling project. The QVT provides the transformation rules because it is a standardized specification for transformations between different models.

Firstly, the root element Schema is selected to be mapped to a Database element in GEDBM using the Schema2DataBase transformation. After, all classes present in a schema are mapped to entities in the target model based on their type. All these classes’ attributes are also mapped using the Attribute2CommonField transformation. Identifier attributes are mapped using the Attribute2IdentifiesField transformation. Finally, the geographical relationships are mapped based on their type, with one transformation for each type in the OMT-G.

C. From GEDBM Metamodel To SFS/SQL Code

The second set of transformation rules is responsible for generating the SFS/SQL code from the GEDBM (which was previously instantiated from a relational or a geographical modeling project). The GEDBM is the source Platform Independent Model from this transformation, while the SFS/SQL is the target Platform Specific Model. We use M2T specification because it generates text artifacts from models. The process begins with the ToSQL transformation, which creates the text file of the generated SQL. Next, the ToDataBase transformation generates the name of the database to be created, and the ToEntity maps all entities to tables in the following order: the entities that do not have foreign keys and the entities that have foreign keys of the generated entities.

The transformation PrintRelationshipAsEntity generates the code for the associative entity. The PrintSelfRelationship does the same for self-relationships. The PrintConventionalEntity generates SQL code, while PrintGeographicEntity, PrintArcNodeTopology, and PrintGeospatialFeature transformations print SFS code. The last transformations generate codes for fields, primary and foreign keys, text and numeric limits, default value, integrity, and check constructions.

D. Discussion

The meta-modeling allows a system construction by defining different abstraction models, starting from a high abstraction model to a particular model. The database project also uses different designing levels, from requirements engineering to a physical project specification using a specific technology. Therefore, using the MDD methodology, it is possible to generate the physical project performing model-to-model transformations from a conceptual model.

The methodology uses a generic metamodel, which allows the use of several relational modeling languages and notations. Once it gathers common concepts from these notations and languages, it can generate graphical representations using a single metamodel. Besides, suppose a graphical representation instantiates (directly or indirectly) the metamodel and exists another graphical representation directly constructed over GEDBM. In that case, it is possible to generate an equivalent graphical representation between these modeling (because the metamodel instance is unique for both modeling in this case).
This feature could be interesting in a database team where the designers do not know or use the same modeling language.

The methodology can also be adapted to other geographical models (e.g., UML-GeoFrame), requiring a new set of transformation rules between the new model and the GEDBM. Then, any existing transformations can be realized from the GEDBM without needing new or modified transformation rules. Likewise, the methodology allows code generation to any other specification (since their concepts are present in GEDBM). In this case, generating new transformation rules for the new specification code is necessary.

IV. DESIGNING A DATABASE USING THE GEDBM

This section presents the developed components of the proposed framework for database modeling. We used GMF to develop the graphical environment; EMF to define and create the metamodels; the M2M and Acceleo to generate text artifacts based on ANSI SQL patterns; and OCL for constraining the models. The environment is available for the Eclipse Framework and is a set of plugins. Figure 3 illustrates the environment for developing diagrams using the Entity-Relationship diagram and the palette of items.
Once a diagram is modeled, it can access its metamodel, where all the structures designed can be found. The EMF tree diagram can add additional information not graphically represented in the model. For example, fields such as Integrity and text limits can be inserted for each attribute. The instantiated metamodel for a Project Management System example can be seen in Figure 4. It is possible to generate the diagram directly from a modeling instance using the metamodel if the user prefers to instantiate the model in the tree view.

Similarly, the specialization concept and the associative entity modeling behavior can be seen in Figure 6. In order to represent the specialization concept, an entity can be a subgroup of other entities allowing the design of a hierarchical structure. The associative entity is referent to exactly two common entities because it is the consequence of a many-to-many relationship. Besides, some concepts are directly instantiated, such as fields and constraints (text limit, numeric limit, default value, integrity, and check).

![Fig. 4. The proposed MDD approach.](image)

The Figure 5 shows a modeling example of the behavior of the metamodel while using the GEDBM tool to create a self-relationship, a ternary relationship, and simple modeling. It is possible to see (the dotted lines) that every rectangle is instanced as an Entity's object while every diamond is instanced as a Relationship. Cardinality is associated to only an entity, but every relationship has at least two cardinalities constraining them. So, it is possible to generate the cardinality for ternary relations and self-relationships, for example.

![Fig. 5. The metamodel behavior for relationships and cardinality.](image)

![Fig. 6. The metamodel behavior for specialization and associative entities.](image)

The OMT-G diagram is modeled using the OMT-G Designer tool [18], a set of plugins for the Eclipse. This tool is independent of our solution and is used to provide geographical modeling in the proposed framework. The tool was developed using an MDD approach with EMF and GMF. The OMT-G Design tool is responsible for instantiating the concepts of the OMT-G metamodel present in the solution. Once this metamodel is instantiated, it is possible to execute the transformation to the GEDBM using a QVT transformation. This process generates a GEDBM model instantiated with the concepts modeled in geographical notation, and an equivalent relational model can be generated. Then, the SQL/SFS code is generated from the GEDBM instance using Acceleo.

The proposed MDD approach avoids human mistakes since it is automatic and assisted by a CASE tool, demanding only more time from the designer in database system early requirements. When designing a system not using an automated approach, some mistakes can be non-intentionally added to the development process. These mistakes are avoided because the transformations from models to models use formal specifications (QVT and M2T) based on OCL. So, if a concept is modeled in early phases, the same concept will be guaranteed to be present in the final database system.

V. RELATED WORK

The CWM uses the MOF to allow the modeling of object-oriented notations such as UML and the XMI. The relational package of CWM covers the SQL standard and relational DBMS concepts. There is no official tool associated with
CWM, but it has several vendors that implement its core. The revised GEDBM allows several relational modeling languages and has an MDA tool for Chen’s notation. It also integrates with the OMT-G geometrical model for modeling geographical concepts. The revised GEDBM does not aim to replace or exceed the CWM. Instead, it could be integrated into the CWM since its core is generic for relational languages. Besides, the GEDBM can be easily mapped to the CWM since it adheres to the MOF, extending the scope for geographical databases.

The EER metamodel aims to provide all concepts for the Enhanced Entity-Relationship model [1], which extends the original ER notation [2]. It is supported by the EERRCASE tool, a graphical tool for Eclipse IDE. The EERRCASE was also developed using the EMF and GMF. The SQL/DDL generation for PostgreSQL and the model validation were constructed using Epsilon Validation Language and Epsilon Generation Language. The EER metamodel claims to be a technologically independent model, but it only provides concepts for Chen’s notation. The GEDBM gathers concepts for several modeling languages. It also provides concepts for some notations, such as MIN-MAX. Besides, the new GEDBM gathers geographical concepts, unlike the EER metamodel. Besides, it does not represent MIN-MAX notation, while the GEDBM does. The EERRCASE tool is bounded to Chen’s notation and generates SQL code for PostgreSQL. The GEDBM allows code generation for the ANSI/SQL, a standard specification for SQL.

The original GEDBM does not gather concepts for geographical models limiting the metamodel scope. In the revised and extended GEDBM, the OMT-G model concepts are added to the original metamodel to enable geographical modeling. Besides, a new set of M2T rules for SQL/SFS is proposed, expanding the metamodel applicability. Besides, the OMT-G Designer was integrated to guarantee a graphical interface for at least one geographical modeling language.

VI. CONCLUSION

This paper presented a revised metamodel for relational and geographical database projects, supported by an automatic code generator for ANSI/SQL and SFS/SQ. The metamodel allows projects to use several modeling languages and notations for relational databases. The geographical part was constructed by mapping the OMT-G concepts directly into the extended GEDBM metamodel. The revised GEDBM aims to fit the gap of generic models for database development since there is no generic model that includes concepts from both geographical and relational models. Besides, current metamodels are bound to specific languages and notations, limiting the designer’s choice of the most appropriate approach.

It also presented an MDA methodology to assist the database project that allows the integration of tools since the metamodel is generic for extant modeling languages. A set of QVT rules was developed from OMT-G to the extended GEDBM. A set of M2T transformations for ANSI/SQL and SFS/SQ was adapted and reused from the original GEDBM to generate automatic codification. It developed a modeling tool based on Chen’s notation, consisting of a set of plugins for Eclipse using the EMF for the revised GEDBM, the GMF for the ER’s visual concepts, and the Acceleo for both cartridges for code generation. The developed framework provides a substantial difference from other ER tools because it is not bound to a specific language or notation and allows geographical modeling. It is also extensible since it was constructed on top of GEDBM. The code generation uses the ANSI/SQL and the SFS/SQ patterns to overcome the technological dependency of certain DBMS.

For future work, it is necessary to apply OCL rules to guarantee the database normalization process and avoid inconsistent models. The OCL assists in the identification of not well-formed modeling before model-to-model transformations or code generation. A reverse engineering Domain Specific Language is proposed using the Xtext tool, which provides grammar constructions for metamodels.

REFERENCES

Expansion Mechanism for Runtime Verification of Self-adaptive Systems

Masaya Fujimoto, Hiroyuki Nakagawa, Tatsuhiro Tsuchiya
Graduate School of Information Science and Technology, Osaka University
Email: {m-fujimoto, nakagawa, t-tutiya}@ist.osaka-u.ac.jp

Abstract—Self-adaptive systems can adapt to environmental changes by modifying their behavior and require runtime verification after adaptation. More efficient verification mechanisms are required because verification mechanisms such as model checking are computationally and memory intensive. A possible method is to generate expressions for model checking at design time and execute such expressions at runtime. Our previous work proposed a caching mechanism and parameterization to improve the expression generation method. In this study, we improve our previous work by generating expressions using Laplace expansion. This method expands the probabilistic model at the points where it is different from the design model and brings the model closer to a model in a cache for generating expressions. We also propose a method to generate candidate matrices to increase the number of cached matrices and improve the cache hit ratio. We conducted experiments with three types of changes, that is, adding, changing, and deleting states. We observed that our approach is effective when the model’s states are added or changed.

Keywords: requirements, self-adaptive system, runtime verification.

I. INTRODUCTION

Software systems are increasingly being used by more people and should be highly reliable regardless of changing operating environments. Self-adaptive systems [1,2,3,4] are systems that can operate stably in changeable environments. A self-adaptive system can restructure itself in response to changes in the external environment, making it easy to change and manage the system. However, a self-adaptive system requires verification at runtime to ensure that the modified system meets system requirements. We focus on the reachability property, which states that a target state can be eventually reached from an initial state. The state space is represented as a discrete-time Markov chain (DTMC) [5] model. This is because many properties useful in software development can be reduced to reachability. Filieri et al. [6,7] proposed a method that places the state transitions that are expected to be changed as variables in advance. This method generates a set of verification expressions that include the variable at design time and assigns obtained parameters at runtime to perform verification. This allows for faster verification. However, Filieri’s method generates a verification expression for system behavior at design time; therefore, it is not possible to perform verification quickly when the behavior changes significantly after adaptation. To address this problem, our previous work [8,9] proposed a method that uses a caching mechanism: Intermediate formulas obtained during the generation of verification formulas are stored in a cache. When the system state changes significantly and recalculation is necessary, the results of the intermediate formulas are reused from the cache, thereby reducing the computation time during system execution. Furthermore, by generating predicted models after adaptation and storing them in the cache as well, the cache hit ratio is improved, further decreasing computation time.

In this study, we attempt to further speed up runtime verification time by improving the caching method in our previous study. Previously, the cache size increased along with the size of the predicted model, resulting in slower execution times. In this study, we use model modification information and perform Laplace expansion from the point where the model state changes when re-generation is necessary. We also generate candidate matrices to improve the cache hit ratio and reduce the calculations done in generating expressions.

The experiment results show that this approach is faster than other methods in specified situation, such as adding states and changing states.

II. RELATED WORK

Previous studies have developed the approach of model checking for runtime verification. For example, [10] proposed a fast parametric model checking (fPMC) approach that generates an abstract model, which represents multiple states with a single state. Thus, even if the model size increases, this method can reduce the computational time. Furthermore, [11] proposed an incremental quantitative verification method for probabilistic models, which re-uses results from previous runtime verification to accelerate the process. The key in this approach is to use a decomposition of the model into its strongly connected

Fig 1. MAPE feedback loop mechanism.
components (SCCs). This method also uses the structure of models and requires the analysis of the change impact on the model before the previous verification. Ref. [12] investigated three techniques, namely caching, lookahead and nearly-optimal reconfiguration. While this technique assumes that the verification is continuously executed, our technique assumes that the verification is divided into at runtime and at design time. A technique of caching is the basis of our approach. A technique of lookahead uses spare CPU to pre-verify stochastic models, which are expected to arise in the future. A technique of nearly-optimal reconfiguration terminates runtime verification as soon as a system configuration satisfies some condition. This paper shows that these techniques can lead to significant reductions in runtime verification.

III. BACKGROUND

In this section we explain self-adaptive systems and the DTMC method of model verification, which is widely used to model the reliability of software systems. In our study, we assume that the system states are represented as models, which enable easy and efficient verification of the reliability of systems in terms of non-functional properties, such as memory consumption and the computational cost.

A. Self-adaptive systems and model verification

The primary mechanism for self-adaptive systems is the monitor-analyze-plan-execute over a shared Knowledge (MAPE-K) feedback control loop [13], which repeats the four steps: monitoring, analyze, plan, and execute (Fig. 1). At runtime, a self-adaptive system monitors its external environment and analyzes information obtained from monitoring. If the system state violates the requirements, the system plans a new behavior that meets requirements and updates itself. This mechanism enables automatic adaptation to the environment. A self-adaptive system requires efficient verification [14] to meet requirements and update itself over time.

B. Discrete-time Markov Chain Model

A DTMC model is defined as state transition augmented with probabilities that meet the Markov process requirement that future states depend only on the current states without depending on previous states. The elements of a DTMC model are as follows:

- $S$ is a finite set of states

- $P$ is a transition probability between states

- $S_0 \subseteq S$ is a set of initial states

- $P: S \times S \rightarrow [0, 1]$ is a transition matrix representing the transition probability between states

A DTMC model has two types of states. The first is an absorbing state, which has transition probability of 1 to itself, while the second is a transient state which has a transition to other states. In this study, the state transition probability is represented as a real value $[0, 1]$ and variables.

Fig. 2 shows a model of how a cleaning robot acts. This model analyzes the information obtained by the sensor and decides actions based on the information. The model transitions to either a failure or success state. The circles in Fig. 2 denote states and each arrow represents a transition from one state to another. The number on each arrow denotes the probability of a state transition. The probability variable is a parameter obtained through execution or expected to change. The system starts at an initial state 0 and transitions to states 1-3, based on information acquired by the sensors. External information is acquired via infrared sensor 1. If infrared sensor 1 has a problem, it transitions to states 1 or 3. In state 4, the system analyzes the information obtained. In state 5, it plans response actions based on the analysis, and transitions to states 6-8. States 6-8 indicate the robot motions; if a movement action is performed without any problems, a transition is made to state 9, which denotes success. Conversely, if the corrective action cannot be performed owing to obstacles, a transition is made to state 10, which denotes a failure state.

A DTMC model can be represented by an adjacency matrix. Fig. 3 shows a matrix representation of the example in Fig. 2. In such a matrix, row i, column j represents the probability of transition from state i to state j.

![Fig. 2. An example of DTMC model verification: a cleaning robot.](image)

![Fig. 3. The transition matrix of Fig. 2.](image)

![Fig. 4. Sub-matrices Q, R, O and I of the DTMC model for the cleaning robot illustrated in Fig. 2.](image)
C. Runtime verification using DTMC model

In this section, we explain how to calculate state transition probabilities for model checking from a DTMC model and system requirements. A DTMC model with absorbing states can be represented by the following four matrices.

\[ P = \begin{pmatrix} Q & R \\ Q & I \end{pmatrix} \]

The matrix \( Q \) is a matrix of probabilities of transitioning from a transient to transient state; the matrix \( R \) is a matrix of probabilities of transitioning from a transient to an absorbing state, and the matrix \( I \) is a matrix of probabilities of transitioning from an absorbing to absorbing state. The matrix \( I \) is an identity matrix because the transition probability to itself is 1. The matrix \( O \) represents the transition probability from an absorbing to a transient state, which is always zero because an absorbing state has only transitions to itself. Hence, matrix \( O \) can be expressed as a zero matrix, as shown in Fig. 4.

Reachability in DTMC model can be expressed by the probability operator \( P_{\text{op}}(l) \), where \( l \) is a path formula. \( \bowtie \) denotes the comparison operator, such as \( <, \leq, >, \) and \( \geq \), and \( p \) is a threshold of the probability that is defined by the requirements. \( P_{\text{op}}(l) \) represents whether the probability meets \( \bowtie p \) under the condition \( l \). We verify whether the probability of reaching an absorbing state satisfies \( \bowtie p \). The following section describes how to obtain the transition probabilities for verifying reachability. To verify reachability in a DTMC, we consider the transition probability from a transient to transient state. If \( Q \) denotes the transition probability from a transient state, the transition probability after the first two transitions can be expressed as \( Q^2 \), which is the product of the first and second transition probabilities. The probability in some steps can be calculated in same way as follows.

\[ N = I + Q + Q^2 + Q^3 + \cdots = \sum_{k=0}^{\infty} Q^k \]

Because matrix \( N \) is an infinite series of matrix \( Q \), matrix \( N \) can be taken as the inverse matrix of \( (I - Q) \). Next, given that matrix \( N \) is the transition probability from a transient to transient state, the reachability can be obtained with the following equation.

\[ B = N \times R \]

Reachability \( b_{ik} \) from an initial state \( S_i \) to an absorbing state \( S_j \) can be calculated as follows.

\[ n_{ij} = \frac{1}{\det(W)} \cdot \alpha_{ji}(W) \]

\[ b_{ik} = \sum_{x=0}^{t-1} n_{ix} \cdot r_{xi} = \frac{1}{\det(W)} \alpha_{xi}(W) \cdot r_{xj} \]

The calculation of \( b_{ik} \) requires the calculation of determinants. The determinant is calculated by Laplace expansion and LU-decomposition.

D. Generating runtime verification expressions

The calculation of determinants is computationally intensive but must be performed at runtime. Therefore, Filieri et al. [5,6] proposed a method by performing some of the calculations required for model checking at design time.

The method consists of two processes: precomputation at design time and verification at runtime. First, precomputation parameters that can only be obtained or may change at runtime are placed as variables, and verification expressions are generated. At runtime, the desired transition probabilities are calculated by substituting parameters into the verification equation to determine whether the requirements are met. This allows for fast model checking even if some of the transition probabilities have unknown parameters. The transition probabilities calculated by this method from initial state 0 to absorption state 10 in Fig. 3 are as follows:

\[ b_{010(x_0,x_1,x_2,x_3)} = 0.02(x_0 + x_1)(x_2 + x_3) + 0.024(x_0 + x_1) \]

At runtime, the reachability property can be obtained by substituting parameter values obtained from sensors and other sources into this expression.

In this method, LU-decomposition is not possible when the matrix includes variables. Therefore, Laplace expansion is performed first; then the variables are removed from the matrix to enable LU-decomposition. This allows for shorter computation times than would have been with Laplace decomposition alone. We denote the size of \( Q \) matrix as \( t \), the average transition number as \( \tau \), and the number of rows including variables as \( c \). To calculate \( b_{ik} \), the calculation of \( t \) determinants that is \( (t-1) \times (t-1) \) sizes of sub-matrices using Laplace expansion is given by \( O(t^3) \). In the case of calculation of determinants with variables, the row including variables is expanded and requires \( \tau^c \) determinants. The expanded matrices are then calculated by LU-decomposition because the matrix has no variables. The calculation of a runtime verification formula is as follows:

\[ O(\tau^c \cdot (t - c)^3) \sim O(\tau^c \cdot t^3) \]

E. Caching mechanism and grouping of states

In this section we describe the caching mechanism in the proposed runtime verification reduction method. Filieri’s method cannot use the generated expression when the system has changed significantly, such as if states have been added and deleted, and needs to re-generate the expression. The re-
calculation of the generated verification formula at design time is time-consuming and affects system performance and execution. To address this problem, a caching mechanism is used. This caching mechanism stores pairs of matrices and the intermediate expressions obtained during the generation of verification formulas and reuses an expression if the matrices match. This is because the changes in self-adaptive systems are partial, and most of the models are similar.

To reduce computational time, this approach generates candidate models and stores their matrix pairs as well. This can enable an early match of a matrix during calculation. Only matrices including the variable are stored in a cache, because a matrix without variables can be efficiently calculated by LU-decomposition. When the size of model matched is large, a correspondingly large computational time improvement is gained from using a cache. The converse is the case when the matched model size is small.

The caching mechanism has a drawback in that the size of the system model increases as the number of stored matrices in a cache becomes large. To solve this problem, in our previous study, similar states are grouped, and the transition of states is limited within the same group. The assumption is that processes can typically be grouped by functions. Hence, processes in the same group are completed in one function and adding of states is restricted to the same group. This can reduce the number of candidates.

Fig. 5 represents the grouping of states example in Fig. 2. States 0-4 belong to sensing group and states 5-10 are in migration group.

IV. EFFICIENT LAPLACE EXPANSION FOR CACHING

The cache mechanism uses intermediate formulas to generate expressions efficiently. As model size increases, cache size also increases. An increasing cache size makes it difficult to perform by the robot, which has restriction of memory usage. Therefore, to improve hit ratio without increasing the cache size, we focus on Laplace expansion. In Laplace expansion, for a square matrix of order n, row i is chosen arbitrarily and coefficient $A_{ij}$ is calculated for each component of the i th row, so that the resulting expansion formula matches the determinant of A. The equation is expressed as follows:

$$\text{det}(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \cdot \text{det}(A_{ij})$$

Laplace expansion can be applied to column j in a similar process by calculating the coefficient $A_{ij}$ for each component in column j to obtain the determinant of A.

To improve the cache hit ratio, we store changes in the model at design time. At execute time, we perform Laplace expansion on the changed points to remove them. Compared to other methods, the matrix does not contain the changed points after adaptation; thus, it is more likely to match the matrices stored in the cache.

Fig. 6 shows an example of how Laplace expansion is performed, and the resulting generated intermediate expression and its matrix are stored in the cache. At runtime, if the matrix does not include any variables, the matrix is calculated by LU-decomposition because LU-decomposition is faster than Laplace expansion. If the matrix includes any variables, the system acquires added rows as model change information and performs Laplace expansion on the rows added during the calculation of determinants. A search is performed on the obtained matrices, and if a matching matrix exists, the matrix is replaced with the expressions corresponding to the matrix.

V. GENERATING CANDIDATE MATRICES

To improve the cache hit ratio, we expand the matrix at design time in different rows, and intermediate expressions are stored in a cache. This increases the number of matrices stored in the cache and improves the cache hit ratio. Reducing the computational time at runtime depends on replacing the calculation of determinants with expressions stored at design time. This requires more matrices correspond with partial changes.

Fig. 7 shows an example of generation of candidate matrices. The third row of the top matrix in the figure is expanded. The expanded matrix and the verification expression from the calculation are stored in a cache. Laplace expansion is done on the fourth row of the bottom matrix in the figure. The resulting matrix is different from the one obtained by Laplace expansion on the third row. The intermediate formulas for these matrices
are stored in the cache as well as efficiently use the model at design time. This increases the number of matrices stored in the cache.

VI. EXPERIMENTS

The experiments compare the calculation time and cache size of the proposed method with those of other methods. In addition, the cache hit ratio is compared; that is the number of matches with the matrix stored in the cache at runtime divided by the number of cache searches. All the programs used in the experiments were implemented in Java. The experiments were conducted using the following methodology.

- State size ranges from 10 to 25 states; the increment is by five states.
- Transition from one state to another is randomly generated within the same group.
- The number of trails is 10, and the average computation time over the 10 trials is used.
- The computation time and cache size required to obtain the probability of transition to absorbing state are measured.

Methods to be compared:

- Baseline method 1: Filieri’s method
- Baseline method 2: Intermediate generative formulas using a cache + candidate model.
- Proposed Method 1: Using Laplace expansion.
- Proposed Method 2: Proposed method 1 + generating of candidate matrices.

Baseline method 1 is the method by Filieri et al. Baseline method 2 uses a cache, generates candidate models at design time, and stores them in the cache. Proposed method 1 uses efficient Laplace expansion. Proposed method 2 performs Laplace expansion at different points at stores the matrices in the cache. We conducted experiments on three types of changes, namely adding states, changing states, and deleting states. The first experiment adds a design model to one state whose transition probability is restricted to the same group. Second experiment changes transition probabilities in one row. The last experiment randomly deletes one state in the design model. We conducted this experiment on a Mac equipped with a 7th generation Core i3(1.2GHz), 8.0 GB RAM, and Java program running on Eclipse.

Tab. I shows the results when the number of states is changed by adding states. The execution times of the proposed methods 1 and 2 show that they can generate runtime verification expressions in less time than other methods. Baseline method 1 does not use a cache; hence, cache hit ratio is 0. Among the other methods, the proposed method 2 has a high cache ratio of 0.61 for 25 states. The proposed method 2 has a smaller cache size than the proposed method 1. Tab. II shows the results when the number of states is changed, and states are partially changed. The execution time of the proposed method 2 is faster than other methods in generating a runtime verification expression, while the proposed method 1 is slower than other methods when the number of states is 25. Despite adding states as well, the cache size and cache hit ratio of baseline method 1 is 0. Tab. III shows the results when the number of states is changed by deleting

### Table I: Results of Experiments on Adding States When the Size of Model Is Changed

<table>
<thead>
<tr>
<th>Execute Time [ms]</th>
<th>Cache Hit Ratio</th>
<th>Cache Size [KB]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline Method 1</strong></td>
<td><strong>Baseline Method 2</strong></td>
<td><strong>Proposed Method 1</strong></td>
</tr>
<tr>
<td>10</td>
<td>13.349</td>
<td>8.824</td>
</tr>
<tr>
<td>15</td>
<td>192.841</td>
<td>133.019</td>
</tr>
<tr>
<td>20</td>
<td>677.729</td>
<td>662.884</td>
</tr>
<tr>
<td>25</td>
<td>2338.353</td>
<td>2278.914</td>
</tr>
</tbody>
</table>

### Table II: Results of Experiments on Changing States When the Size of Model Is Changed

<table>
<thead>
<tr>
<th>Execute Time [ms]</th>
<th>Cache Hit Ratio</th>
<th>Cache Size [KB]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline Method 1</strong></td>
<td><strong>Baseline Method 2</strong></td>
<td><strong>Proposed Method 1</strong></td>
</tr>
<tr>
<td>10</td>
<td>13.752</td>
<td>11.17</td>
</tr>
<tr>
<td>15</td>
<td>389.014</td>
<td>270.952</td>
</tr>
<tr>
<td>20</td>
<td>3451.744</td>
<td>3472.968</td>
</tr>
<tr>
<td>25</td>
<td>15957.526</td>
<td>17634.146</td>
</tr>
</tbody>
</table>

### Table III: Results of Experiments on Deleting States When the Size of Model Is Changed

<table>
<thead>
<tr>
<th>Execute Time [ms]</th>
<th>Cache Hit Ratio</th>
<th>Cache Size [KB]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline Method 1</strong></td>
<td><strong>Baseline Method 2</strong></td>
<td><strong>Proposed Method 1</strong></td>
</tr>
<tr>
<td>10</td>
<td>1.792</td>
<td>2.412</td>
</tr>
<tr>
<td>15</td>
<td>26.777</td>
<td>19.018</td>
</tr>
<tr>
<td>20</td>
<td>312.608</td>
<td>328.058</td>
</tr>
<tr>
<td>25</td>
<td>1333.708</td>
<td>1695.101</td>
</tr>
</tbody>
</table>
states. The execution time of method 1 is the fastest and proposed method 1 is the slowest in these experiments.

VII. DISCUSSION

A. Experiments on adding states

In the experiments on adding states, proposed method 2 is faster than other methods in generating a runtime verification expression. Efficient Laplace expansion enhances the cache hit ratio and leads to reduction in computational time by replacing the calculation of determinants with expressions. In particular, the cache hit ratio of proposed method 2 is stable owing to large size of matrices in the early phase, whereas the cache hit ratio of the other methods decreases.

B. Experiments on changing states

Tab. II shows the execution time for "changing states" experiments. The execution time of the proposed method 2 is less than those of other methods; however, the proposed method 1 takes the longest time to generate an expression. This is because the number of intermediate expressions stored with proposed method 1 is smaller than the number stored with proposed method 2. The cache of proposed method 1 does not include the matrices in cases of partial model changes. This leads to waste of cache search and increases computational time. Compared to adding states, as the states increase, the cache hit ratio of proposed method 2 is lower because of having to match a small number of matrices.

C. Experiments on deleting states

Tab. III shows the results for deletion of state. Our proposed method is not effective in deletion of states, method 1 is the fastest, and the proposed method 2 is the slowest of the methods. Because proposed methods 1 and 2 do not have enough matrices (which corresponds with deleting states), search misses increase execute time. The cache hit ratio of the proposed method 2 is higher than those of the other methods, because cache hit locally is much higher. We should consider improvement to increase the number of matrices in deleting states situations.

These experiments demonstrate that our approach is effective in adding and changing states but is not effective in deleting states. Additionally, we found that the size of the matrices matched affects the execution time. In future work we will devise a method that matches as large a size of model as possible. We would decrease the cache size to increase the applicability of this method.

VIII. CONCLUSION

In this study we described a runtime verification mechanism for self-adaptive system. We propose the method using efficient Laplace expansion for caching. We also generate candidate matrices to increase storage of intermediate runtime verification expressions when significant changes occur, such as adding, deleting, and changing of states. These approaches improved the cache hit ratio by expanding the changed points of the matrix and reduced the computational time for generating runtime verification expressions. This led to fast runtime verification. The proposed method, which generates candidate matrices and expands the changed points, is effective when the states can be grouped and the model undergoes changes, such as adding or changing states. A possible application of the proposed mechanism is dynamic web applications.

In future research, we aim to improve this method with experiments on deletion of states and partially changing states. Additionally, we will consider different adaptation patterns, such as adding more states and changing the number of variables to extend the applicability of the caching mechanism.

ACKNOWLEDGMENT

This work was supported by JSPS Grants-in-Aid for Scientific Research (No.20H04167).

REFERENCES


Automatic Discovery of Controversial Legal Judgments by an Entropy-Based Measurement

1st Jing Zhou  
Center for Applied Statistics and School of Statistics  
Renmin University of China, China  
jing.zhou@ruc.edu.cn

3rd Fang Wang*  
Data Science Institute  
Shandong University, China  
wangfang226@sdu.edu.cn

2nd Shan Leng  
Department of Statistics  
University of Wisconsin-Madison, USA  
shan.leng@wisc.edu

4th Hansheng Wang  
Guanghua School of Management  
Peking University, China  
hansheng@pku.edu.cn

Abstract—The judgment of controversial cases has always been an important judicial issue, but it is not easy to discover them in practice. In this paper, based on 1,361,354 legal instruments data collected from China Judgments Online, we adopt a deep learning framework to classify 147 different kinds of crimes. The proposed method has three critical steps: 1) We adopt a deep learning model to predict crime categorization; 2) With the trained model, each case is given a score vector which represents the probability that it belongs to each crime; 3) With the probability score, we develop an entropy-based index to measure the controversy of each case. We find that the larger the entropy, the more inconsistent the result given by the model based on the first instance judgment. To verify the proposed entropy measure, we provide 1) two-sided evidence based on second instance judgments; 2) comparison with some baseline models. Both confirm the practical usefulness of the entropy measure. Our results indicate that the proposed framework has an ability to discover potentially controversial cases. It should be noted that the goal of this study is not to substitute the model result for the judge's decision, but to provide a guiding reference for the judicial practice of sentencing.

Index Terms—legal judgment prediction, controversial case discovery, entropy-based measure

I. INTRODUCTION

The criminal law is one of the most important foundations of a nation’s legal system. It grants the most powerful protection to every person’s right to access to justice. Good practice of criminal law is beneficial to national security, social stability, economic development, and tranquility. Otherwise, justice can scarcely be realized in practice. However, enforcing criminal law is not a trivial task. This is particularly true for a country with a huge population and a large number of legal cases like China. According to the statistics from the Supreme People’s Court, 1.12 million first instance cases were judged by courts in 2020 ¹. Many of those went through a second instance (about 3.73%). This suggests that a large amount of litigation was not terminated (no appeal or protest). These are the controversial cases, that is, litigation where judges might hold very different opinions. Ideally, controversial cases would not occur because different judges should come to the same judgment. However, since China’s legal system is statutory, judges often have different opinions on the same or similar cases due to the ambiguity or uncertainty of written legal provisions, which leads to controversial rulings.

The discovery of controversial cases is critical to legal system development. Controversial cases are misjudged in the first place sometimes. To bring them to the attention of the highest-level judge can maximize their likelihood to be rectified, leading to better justice in practice. Even for correctly judged controversial cases, it is still of great importance to bring them to the attention of legal practitioners so that cases of similar types can be better judged in the future. But the identification of controversial cases is not easy. Consider, for example, Among the 1.12 million criminal first instance cases judged in China in 2020, a nontrivial number of cases warrant attention. The question is how to discover them.

We propose a deep learning framework for the automatic discovery of controversial cases. The proposed research framework is based on two components. The first is the database. As mentioned before, since 1996, every trial case in China has been documented in text format, and most of these are published on China Judgments Online. We developed a database of 1,361,354 trial records in text documents, covering 268 kinds of legal judgments occurring in China in 2018. A text document usually includes details such as the facts, the court’s opinion, and the verdict. This constitutes the data foundation. The second basis is deep learning models. Since the trial cases are recorded in natural language, the capability to understand natural language is of great importance. In this regard, we benefit from development in deep learning related natural language processing skills. A number of important models are studied and further modified so that they can work better for controversial cases discovery.

We are not the first to use deep learning models to study legal problems. Various researchers have used this technique

DOI: 10.18293/SEKE23-035  
*Corresponding author

¹http://www.court.gov.cn/zixun-xiangqing-290831.html

82
to explore the issues of legal judgment prediction (LJP) [1], [10]. Some studies try to extract criminal elements or domain concepts from legal documents to predict legal results [3], [8]. For example, [6] investigated several discriminative legal attributes to improve the accuracy of low-frequency charges. Other studies focus on jointly modeling multiple subtasks by some novel multi-task legal judgment prediction frameworks in LJP [15], [16], [17] proposed a TopJudge model, which made use of the relationship between subtasks in LJP. [12] used a multiscale attention mechanism in charge prediction with multi-defendants. Another group of researchers has focused on similar case matching [14]. They concern the finding of pairs of similar past cases. Typical research topics are information extraction [7] and similarity calculation [14]. To calculate similarity at the semantic level, information such as citations [13] and legal concepts [11] are used. Recently, [2] proposed a heterogeneous graph embedding method for Chinese legal document similarity measure. Other issues include legal question answering and legal assistant systems. [18] provided a detailed literature review of the use of deep learning methods for legal problems.

To summarize, we highlight the following contributions. The quantitative relationship between facts and crimes is first extracted by a deep learning model. Second, based on model results, we propose an entropy-based index to measure the controversy of cases, leading to the discovery of controversial results, which can provide a basis for justice, as well as guide the evolution of criminal law and theory.

II. LEGAL JUDGEMENT DATA

A. Data description

The data were collected from the China Judgments Online website2, which contains the public legal instruments of criminal cases occurring in mainland China in 2018. There are 1,361,354 legal instruments covering 268 types of crimes. Each legal instrument contains various variables, where the court investigation is the most important information. It represents the opinions of the judge, which are supported by facts and evidence. The court investigation includes the constitution of the crime and circumstances for sentencing that must be considered at the trial. For example, the crime of murder must include elements such as intention, action, results, and capacity of criminal responsibility. Sentencing must consider surrender, confession, and other circumstances, which should be clearly written. To this end, we extract two key parts from the court investigation, which consist of text content beginning with “verified by trial” and “this court held that.” We aim to find facts verified by the court from the first part of the text, and information about crime constitution and sentencing circumstances from the second part.

B. Data preprocessing

The original dataset is noisy, and preprocessing is required. We follow the next four steps to exclude some samples. First,

we exclude legal instruments that do not contain the final judgements. Second, we focus on first instance cases, i.e., the first trial for a case. Third, we eliminate legal instruments that involve multiple crimes or defendants. Finally, crimes with very small sample sizes (e.g., less than 30) are also excluded. The final sample used for crime classification contained 731,454 legal instruments, covering 147 crimes.

III. DEEP LEARNING MODELS FOR CRIME CLASSIFICATION

A. RNN and LSTM models

Before introducing the model structure, we formulate the crime classification problem as follows. Let \( X_i \) be the description of the \( i \)-th judgment, \( 1 \leq i \leq N \), where \( N \) is the number of judgments, which can be represented by a word sequence as \( X_i = \{ X_{it} : 1 \leq t \leq T_i \} \), where \( T_i \) is the length of \( X_i \), and \( X_{it} \) is a word generated from a fixed vocabulary \( W \). Recall that we extracted text content beginning with “verified by trial” and “this court held that” as our target for analysis. In this case, the vocabulary is constructed as all the unique words obtained using word segmentation for that target content. Next, let \( Y_i \) be the associated class label (i.e., the crime type). Then, using a deep learning model, we wish to predict \( Y_i \) by \( X_i \). Remarkably, fact descriptions \( X_i \) might have different lengths as a word sequence. So we calculate the length of each \( X_i \) and their percentiles, and find that about 98%-99% of the text information in the original content can be retained if the maximum length of the word sequence is set to 1,000. Therefore, we use this as the maximum length of a word sequence and fix \( T_i \) at 1,000. For sequences with \( T_i \) less than 1,000, we do padding to achieve that length.

We next consider how to construct a classical model based on \( X_i \). Theoretically, we should map each keyword in the vocabulary to a vector with a high dimension \( d \) [9]. We choose \( d = 128 \) and obtain a sequence of vectors \( Z = f(X_{it}) \) for some mapping function. We consider how to predict \( X_{i(t+1)} \) by incorporating information from both current text and historical states such that the semantic information from the entire fact description up to the \( t \)-th keyword can be fully used to predict \( X_{i(t+1)} \). We hope to establish a functional relationship between \( X_{it} \) and \( Y_i \). Hence an RNN [4] model is constructed, with eight layers: one input layer of text content with dimension 1,000; one embedding layer with 128 hidden nodes; one simple RNN layer with 64 hidden nodes; one global max pooling layer; two dropout layers; and two fully connected layers with dimensions of 512 and 147 respectively. This model has 57,047,123 parameters in total.

The LSTM model is a more improved model to balance long- and short-term dependencies [5] compared to RNN. Our LSTM structure is similar to that of an RNN, with the RNN layer replaced by an LSTM layer with 64 hidden nodes. This gives the LSTM model 57,084,179 unknown parameters.

B. Training process and results

As for training of the proposed RNN and LSTM models, we randomly divide the data into training and validation sets at an 80:20 ratio. To minimize the loss function, a standard
The mini-batch gradient descent algorithm is utilized with a mini-batch size of 256. The learning rate is determined as 0.001 and the Adam optimization algorithm is applied.

To obtain stable results, we train each model for 50 epochs. The loss curves on the training and validation sets for the two models are presented in the left panel of Figure 1, and the prediction accuracy curves are shown in the right panel. The final prediction accuracy on the validation set is about 0.9787 for RNN and 0.9795 for LSTM. It can be seen that there is little difference between the two models in terms of accuracy. This suggests that deep learning models have a certain ability for crime classification.

C. An entropy-based measurement

To define an appropriate measure for the discovery of controversial cases, we consider two representative cases. One is the crime of illegally transferring or reselling land use rights. For this case, the top five crimes predicted by both RNN and LSTM are illegally transferring or reselling land use rights; illegal acquisition, transportation of illegal logging, and deforestation; purchasing abducted women and children; privately dividing state assets; and destroying computer information systems. Their predicted probabilities are respectively 20%, 10%, 7%, 6%, and 5%. As one can see, both RNN and LSTM are confused about which crime it should be classified as, because no crime’s predicted probability dominates the others. This seems to be a strong indication of controversial cases, at least for RNN and LSTM models. In contrast, both RNN and LSTM correctly predict the crime of illegal medical practice with nearly 100% probability, strongly indicating that both models are affirmative about this crime, which therefore is less likely to be controversial.

From the above discussion, we find that if the predicted probability are concentrated on one category, then this is a case of clear judgment, and little controversy should be found. In contrast, a controversial case should involve at least two crime types, and should confuse both algorithms. Accordingly, their predicted probabilities should be comparable, i.e., no crime type’s predicted probability dominates the others. Therefore, a measure to quantify how the predicted probability is distributed across different crime types might be useful to discover controversial cases. This suggests an entropy-type measure for the i-th judgment, \( H_i = -\sum_{r=1}^{R} p_{ir} \cdot \log(p_{ir}) \), where \( p_{ir} \) is the probability that the i-th judgment belongs to the r-th crime type. Based on this, we can conclude that the larger the entropy value, the more dispersed the predicted probability distribution, and the more likely that a judgment will be controversial.

We next apply this measure to the validation set, which leads to a total of 146,291 entropy scores. All the judgments are sorted by descending entropy scores, and are divided into 10 equal-sized groups. For each group, we report the misclassification rates of the RNN and LSTM models in Figure 2, from which we find that the first group has the largest misclassification rate, which is 20.06% for RNN and 19.02% for LSTM. This is expected because cases with large entropy scores should be the most difficult to predict. Accordingly, these cases are the most confusing for both deep learning algorithms and human experts.

D. Comparison with other methods

To further verify the practical usefulness of the proposed entropy-based measure, we first present an analysis based on second instance judgments, i.e., second trials by a higher court based on an appeal or a prosecutor’s protest. Such a verdict or judgment is a strong legal testimony, indicating that the judgment of the first instance might be controversial. We expect that if a first instance judgment with large entropy value is controversial, then its second instance is more likely to rescind the original judgment and remand the case to the original court or directly change the conviction. Specifically, we present two-sided evidence to support the proposed entropy
score measure. First, we calculate the number of cases with an appeal or protest in the first group, as shown in Figure 2, which is 275, accounting for 10.24% of the judgments in this group. However, the corresponding proportion is only 3.73% in the whole dataset. Second, for those 275 second instances, the number of cases remanded to the original court for retrial, or whose original conviction is directly changed, is 248, accounting for 90.18% of the second instances, while in the whole dataset this ratio is only 13.1%. Both confirm that the proposed entropy measure should be of great practical use.

Moreover, we compare our method with some baselines. An intuitive baseline could be a model that directly predicts whether a case is controversial. So a binary classifier predicting whether a case is controversial or not is developed. Such a classifier can be trained over the text of cases regarding whether they are appealed against or not. We use LSTM, RNN, Bert, and Transformer to train the binary classifier respectively. For each baseline, we apply the best model to the validation set to compute a probability score (i.e., how likely it is appealed against) for each judgement. All the judgements are sorted by descending probability scores, and divided into 10 equal-sized groups. For each group, we calculate the appeal rate. Table I displays the results in the top group for each baseline. We can see that the best baseline is Transformer with the largest appeal rate of 18.6%, which is lower than the misclassification rate obtained by our method (i.e., 20.06%). These direct binary classifiers do not perform as well as the proposed method.

<table>
<thead>
<tr>
<th>Model</th>
<th>LSTM</th>
<th>RNN</th>
<th>Bert</th>
<th>Transformer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appeal Rate</td>
<td>11.14%</td>
<td>12.59%</td>
<td>5.00%</td>
<td>18.60%</td>
</tr>
</tbody>
</table>

IV. Conclusion

Based on a large amount of judicial document data, we proposed an entropy-based measure to discover controversial cases. Our results can assist judges to identify potentially controversial cases. However, there are still some limitations. First, we removed charges with small sample sizes during preprocessing, which may be treated as rare events worthy a separate study. Second, we only considered RNN and LSTM models, to the exclusion of more advanced deep learning models. This should be another direction for future research. Finally, we did not consider the problem of sample imbalance, which may bring some issues in the classification. It would be of great interest to solve this problem.

V. Acknowledgement

Zhou’s research is supported in part by the National Natural Science Foundation of China (No. 72171226, 11971504), the Beijing Municipal Social Science Foundation (No. 19GLC052), the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China, No.21XNA027. Fang’s research is supported in part by the National Natural Science Foundation of China (No. T2293773), Hansheng’s research is partially supported by the National Natural Science Foundation of China (No. 12271012, 11831008) and the Open Research Fund of the Key Laboratory of Advanced Theory and Application in Statistics and Data Science (KLATASDS-MOE-ECNUKLATASDS2101).

REFERENCES

Context, Content, Consent – How to Design User-Centered Privacy Explanations

Wasja Brunotte*, Jakob Droste†, Kurt Schneider*
*Leibniz University Hannover, Software Engineering Group, Hannover, Germany
†Leibniz University Hannover, Cluster of Excellence PhoenixD, Hannover, Germany

Email: {wasja.brunotte, jakob.droste, kurt.schneider}@inf.uni-hannover.de

Abstract—In the context of the ongoing digitalization of society, human values such as privacy, ethics and trust are becoming increasingly important. Digital systems are entering private and professional spaces, which in turn affects the privacy of their end users. Hence, there is a need for conveying privacy information in a transparent and understandable manner, with the user in the focus. Lawmakers introduced privacy policies as a means of communicating privacy information. However, those documents have proven to be practically useless for end users. Privacy policies are long, vague, ambiguous and use complex language, such as legal terms, which often require profound background knowledge. Explainability has shown potential as a means to increase transparency and foster trust in software systems. Based upon the foundation of explainability, we developed a layered concept for user-centered privacy explanations, which is implemented within a high-fidelity software prototype. Finally, we tested and evaluated our concept by conducting an interactive user study with 61 participants. The results of our study suggest that our layered design concept enabled participants to understand the privacy aspects they regarded as important. We conclude that our approach seems to be an appropriate way to communicate complex privacy information to end users.

Index Terms—Privacy, Privacy Explanations, Explainability

I. INTRODUCTION

Software systems accompany and support us in our everyday lives, e.g., at work, when consuming information, when purchasing or distributing goods, and when keeping in touch with friends. Human values such as accessibility, ethics, privacy, transparency, and trust are playing an increasingly important role for software engineers [1]. Due to the great advantages that digital systems offer us, users tend to forget that they are not just consuming information. Users reveal a lot of private information about themselves when interacting with digital systems. Often this happens without explicit knowledge and consent of the users [2], [3].

In an effort to protect end users and their online privacy, the European Union has introduced the General Data Protection Regulation (GDPR) in 2018. Article 12 of the regulation [4] states that the processing of personal data must be explained “in a concise, transparent, intelligible and easily accessible form, using clear and plain language”. However, this is at odds with the form in which privacy information is usually encountered in today’s digital spaces. On the one hand, we have privacy policies that are long, unreadable and purposefully opaque [2], [5]. On the other hand, there are short form privacy notices such as cookie banners, which are unspecific, unclear and were found to employ dark design patterns, aiming to confuse end users [6].

Software providers are legally and morally obligated to provide end users with accessible and suitable explanations on how their personal data is processed. However, they do not have the means to achieve this at the current point in time. Hence, there is a need for ways to convey privacy information in a form that is both understandable and satisfying for end users. To bridge this gap in research, we propose privacy explanations since they may have the potential to remedy this problem. Furthermore, Brunotte et al. [3], [7] have shown that they can foster end users’ trust in software systems. This could also be an incentive for commercial software providers to employ privacy explanations, as an increased end-user trust might be in their corporate interest and might lead to more customer loyalty as well as a positive company image [3].

Within this work, we build upon our previous research [3] and investigate privacy explanations through the lens of explainability. The goal of this work was to investigate how to design user-centered privacy explanations that meet the needs and expectations of end users. To this end, we adapt two types of explanations from the field of explainable artificial intelligence (XAI) - contrastive explanations and example-based explanations - and embed them within a layered structure for privacy explanations. We evaluate our design within an interactive user study with 61 participants, which includes the navigation of a prototypical implementation of those concepts. Our results suggest that our design is able to effectively and appropriately convey privacy information to end users. Furthermore, they support the notion that privacy explanations have a positive influence on end user’s trust in the software system as well as on their privacy awareness and underline the importance of context, content, and consent.

II. BACKGROUND AND RELATED WORK

Online privacy “is about an individual’s control over their personal information in virtual space and their right to withhold this information” [3]. Individuals should have the right to know by whom their data may be accessed and also at what point in time this occurs. It is precisely this gap that privacy explanations address. The current state of the art and, thus, the primary channel to inform users about the data practices of a service provider, are documents such as privacy policies or short privacy notices. However, these kind of documents are insufficient to inform users with respect to their privacy. They are too long, too vague, and too hard to understand [2], [5]. As a result, these documents are often ignored by end users. Privacy explanations can be an alternative to educate the user in a transparent and comprehensible way. In light of this, a privacy explanation is not a document such as a privacy notice, privacy statement, or privacy policy. Instead, it is literally an explanation regarding a certain privacy aspect. This means that a privacy explanation is “a piece” of information that is given by a system in a specific context to an addressee (e.g., end users), with the aim of informing and educating the addressee about a specific privacy aspect. Due to space limitations, we refer the reader to [3] for our formal definition.

III. RESEARCH GOAL AND DESIGN

The goal of this work was to investigate how to design user-centered privacy explanations that meet the needs and expectations of end users. Data from a previously conducted survey [3] formed the

DOI reference number: 10.18293/SEKE2023-032
basis, which we validated and extended through a literature review (LR) in the area of explainability and privacy. By means of the gathered data, we conceptualized and refined our vision of user-centered privacy explanations and developed a high-fidelity prototype, which we evaluated through an interactive user study. The focus of this paper is on the results obtained through the qualitative research we conducted. Due to space limitations, we refer to our supplementary material for more details on our approach, especially the conduction of the LR, validation process as well as the user study [8]. Our research was framed by the following research questions (RQs):

**RQ1:** What relevant information should a privacy explanation contain to meet the needs of end users?

**RQ2:** What are appropriate strategies for designing user-centered privacy explanations to convey the relevant information?

RQ1 focuses on investigating what information is required to inform and educate end users. Here, it is important to consider the relevant information that, on the one hand, fits into the context of current data use (scope), and on the other hand, meets the expectations and needs of the users. One challenge is to present satisfactory information without confronting the user with long texts, similar to a privacy policy. By the means of RQ2, we want to assess how to design user-centered privacy explanations. We need to investigate implementation strategies of how to incorporate privacy explanations into a system. Therefore, it is necessary to translate the high-level requirements into concrete design choices.

**A. Research Design**

1) **Literature Review - Validation of Survey Data:** In our previous work [3], we conducted an online survey to identify the need for privacy explanations. The data resulting from this survey forms the foundation of our concepts. The data revealed that, in addition to privacy, the NFRs transparency, understandability, trust, and trustworthiness in particular stood out as critical for privacy explanations. For this work, we supplemented the previously conducted survey with a thorough LR in the area of explainability and privacy, with respect to the aforementioned NFRs. The aim of this step was to validate the data we obtained against literature and to broaden our knowledge.

The starting set for our LR originates from an LR in the field of explainability by Chazette et al. [9]. Our work extends this LR by conducting additional steps of snowballing as well as a data base search. Both with a focus on the relationship between privacy and explainability. We scanned the final set from Chazette’s SLR for relevance concerning the NFRs explainability, privacy, transparency, trust, trustworthiness as well as understandability, and conducted our snowballing process in accordance with Wohlin’s guidelines [10].

2) **Conceptualization & Prototyping:** We approach our concepts for user-centered privacy explanation in two steps. First, we collated and analyzed our data from the previous step in order gain a deeper understanding of what are relevant information and privacy aspects for providing satisfactory and understandable privacy explanations. Here, we identified the importance of context, content, and consent to which we continue to refer to as the 3C-principle. Second, we conceptualized these findings within our design for privacy explanations and incorporated them into the conception of the high-fidelity prototype.

3) **User Study Design:** To evaluate our concept, we conducted an interactive user study with 61 participants. We invited our personal networks to participate in and share our study. Subsequently, we made appointments with those who replied positively. The study consisted of three main sections. It was provided in the form of an online survey, which included two questionnaires and an interactive section in the middle, which made use of a high-fidelity privacy explanation prototype. The prototype was designed in accordance with our previously described concepts. All participants were asked to use the “Think-Aloud” approach throughout the whole study. While completing the questionnaire and interacting with the prototype, they were encouraged to comment and reason their decisions. The results of the coding process of participants remarks can be found in our supplementary material [8].

**IV. RELEVANT INFORMATION FOR PRIVACY EXPLANATIONS**

**Requirements for Privacy Explanation.** As end users come with all kinds of different knowledge and backgrounds, they have different needs and attitudes when it comes to privacy. In order to provide meaningful privacy explanations, we must first understand what privacy aspects are required to be explained. Privacy explanations essentially pursue two abstract quality goals: what information is legally required and what information is required by users. From these superordinate goals, further requirements can then be refined. We were able to elicit a set of high-level requirements in our previous work [3], which we confirmed through our LR. We found evidence in the literature that the four clusters of the requirements are justified: Data Usage, Data Storage, Confidentiality, and Presentation. The cluster Data Usage educates a user what, why, and how data is collected [11]. We summarize these two W-questions and one H-question as the 2WH principle. It was also found that users attached importance to the question “what happens in case of non-consent?”. Data Storage comprises information about where and how long the data is stored [4]. Information about the deletion, and if safeguards are taken into account, is also included in this cluster [3]. Safeguards refer to precautionary measures such as encrypted storage of data, access restrictions, etc. Confidentiality provides information on, among other things, who has access to the data and whether the data is resold [11]. The Presentation Form determines what a privacy explanation should look like, both on a linguistic level and in terms of the medium (textual, visual, audio) [9].

**Context, Content, Consent – 3C-Principle.** The analysis of the literature revealed that privacy explanations can essentially be embedded in three key points: context, content, and informed consent (3C-principle). In general, context is characterized by a specific situation involving a person, a system, a task, and an environment [9]. In the case of privacy explanations, we speak of context when a system processes, requests, or obtains explicit or implicit permissions (e.g., a smartphone app needs access to the camera) to perform a specific task. With reference to our definition, a privacy explanation is always context dependent (contextual) [2], [12]. Thus, a system should contextually supply its user with the needed information – the privacy explanation.

The content is aligned with the context and thus relates directly to it. Content plays an essential role in terms of the effectiveness of the information presented and is framed by the design and structure of the presented explanation (presentation form). This implies that the content is also shaped by the needs of the respective end users. With respect to privacy explanations, the content covers the requirements cluster data usage, data storage, and confidentiality.

When, in a given context, a privacy explanation is presented to the user, it is done with the aim of informing and educating users about the use of personal data in the given situation. This should
enable them to make an informed consent about the use of their personal information [5]. A privacy explanation should not make use of so-called dark patterns such as forced consent, as that would be in complete contrast to the actual purpose.

**Answering RQ1:** End users want to be informed about privacy matters in accordance with the 2WH principle. They also expect to learn whether they will face any disadvantages if they do not consent to data use. Information about the storage of data is also considered relevant. Providing users with this relevant information, according to the current context, puts them in the position to give informed consent, respects their right to self-determination, and might even foster their privacy awareness.

V. **Designing User-centered Privacy Explanations**

In the following section, we will build upon our findings concerning RQ1 to propose appropriate strategies for designing user-centered privacy explanations. Furthermore, we answer RQ2 by discussing the results of our user study, in which our concept was evaluated.

**Conceptualization - A layered Approach.** In order to effectively inform end users, we need to provide them with an explanation complexity that fits their background knowledge and needs. In this context, previous works have argued for the potential of personalized [13] or layered explanations [3], [5]. For our presentation form, we chose to provide privacy explanations in a layered manner. Note that the layers do not build upon each other. Instead, each layer contains its own set of privacy aspects. This way, all necessary privacy information is readily available and accessible, and users can decide for themselves which parts they want to read, without being overloaded with information. Using our defined requirements for privacy explanations as a basis, we introduce five layers of privacy explanations. Each layer covers different privacy aspects, and altogether, they include all necessary privacy information.

The first two layers of the privacy explanation are focused on data usage, using regular text explanations and contrast. Within the first layer and in accordance with the 2WH principle, end users are told what data is used, why it is used and what happens in case of non-consent. This baseline explanation is critical for end users’ understanding of how their data would be used and enables them to decide, whether or not to consent to that processing. The second layer of the privacy explanation is a contrastive explanation, which, as the name implies, stands in contrast to the baseline explanation. Contemporary explainability research has found contrastive explanations to be an effective way to convey information to end users [12]. Within the contrastive layer of the privacy explanation, we tell users in which ways their data will not be used, if they choose to permit processing.

A proven presentation form that can provide additional context to end users are example-based explanations. Adadi and Berrada [12] state that “amongst agnostic methods, visualization is the most human-centered technique”. Within the third layer of our privacy explanation, we opt to provide example-based explanations. Ambiguous explanations about how data is being processed can lead to decision that do not suit end users’ actual privacy preferences [14]. Providing a visual example should be an efficient way to solve this issue. Furthermore, by choosing examples that fit the software system and its typical use cases (contextual), we can provide additional context for the end user. Within the forth and fifth layers, we provide information on additional privacy aspects for interested end users. In essence, these layers are covering details on data storage and confidentiality, as described in our previously defined requirements for privacy explanations. The forth layer explains the circumstances of data storage and the rights of the end users. The fifth and final layer consist of a third party explanation, which lists all third parties who would gain access to the data and briefly states how they might process it [4].

**Results from the Study.** When comparing the different types of explanations with each other, we employed the Wilcoxon Signed-Rank test to test for statistical significance. With the exception of the example-based explanation, most participants (85%) perceived all layers of the privacy explanation to be relevant to their needs. In particular, participants evaluated the example-based explanation as less relevant than the baseline explanation (statistically significant with \( z = 4.16404, p < 0.00001 \)). While every other explanation layer was seen as important on average, the examples were only moderately important to the average participant. While “thinking aloud”, 15% of participants commented that they thought the examples were not necessary, as they added no new information, but only provided another presentation form for privacy aspects that were already explained. In the vast majority of cases, participants had no problem understanding the privacy explanations.

In their “Think-Aloud” comments, 26% of participants attributed this to the explanations being short and concise. 39% highlighted a positive effect of the layered approach on the explanations’ understandability. Both the contrastive and the example-based explanations have shown to be suitable forms of privacy explanations, when it comes to their understandability. In particular, the example-based explanation was regarded as more understandable than the baseline explanation (statistically significant with \( z = -2.53252, p < 0.01 \)). Furthermore, 43% of participants specifically commented on the importance of providing helpful examples. 30% of participants highlighted the importance of the contrastive explanation, remarking that it addressed some of their pre-existing privacy worries.

**Answering RQ2:** Concerning the design for privacy explanations, we find that successful strategies comprise proper explanation types, using suitable presentation forms, within a layered structure. Both, the contrastive and example-based explanations have shown to be understandable and relevant to our participants. Privacy explanations should stay short and concise. This can be supported by the use of visual examples. The layered approach was successful in providing our participants with privacy information in a manner that suits their needs. The vast majority of participants found the explanations to be understandable. We attribute this to our layered approach, that broke up the large amount of information and provided structure to it.

VI. **Discussion and Threats to Validity**

The first contribution of this work is a comprehensive overview of relevant information that should be included in a privacy explanation. This information frames the needs an expectations of end users regarding their privacy and is aligned with our identified 3C-principle, which shapes our design concept for user-centered privacy explanations. Context in which privacy explanations are given, as well as their timing, are of crucial importance [9] and already embedded in their definition [3]. Making use of examples is a prime opportunity to provide context, as they can show how a specific application would process data in a common usage scenario. Notably, the results of our study have shown a discrepancy between example-based explanation’s perceived importance and understandability. Even though the information provided via the examples was seen as the least important, it was perceived as significantly more understandable to participants than the textual explanations. In accordance with [12], we conclude that using visual examples, and thus providing context, is indeed an effective way to explain privacy information to end users. However, the examples need to be carefully chosen and should not
include redundant information if possible. According to the GDPR and backed by the findings of our LR, the contents of privacy explanations should mainly be characterized by answers to the 2WH questions, and should be enriched with non-consent information as well as information on the storage and retention of private user data. Information about who has access to their data and whether it is aggregated or resold might also be considered important by end users. It is important to note here that not all information is equally relevant for every user, as different users have different privacy attitudes [3].

Our second contribution are our strategies for designing for user-centered privacy explanations. The results of our study suggests that the layered approach is a suitable solution to the issue, as it addresses the challenge of providing appropriate information granularity for the different needs of end users. Doing so, we can meet the different needs, expectations, and attitudes of different end users. We refer to this as achieving informational completeness. Following this line of thought, we call our concept user-centered inclusive design, since it does not exclude any user and takes into account their respective privacy attitudes. The results of our research on privacy explanations suggest that they contribute to better understanding, inform and empower users. Thus, privacy explanations can lead to informed consent by the end user, which in turn has a positive impact on their privacy awareness, because users are sensitized through education. In light of this, genuine informed consent is only possible if both context and content are appropriately supplied [15], and this is exactly the approach we would like to address with our proposed user-centered inclusive design.

A. Threats to Validity

Despite careful planning and execution of our research approach, there are still some threats to validity regarding our obtained results. Literature Review. The review process requires a common understanding of the methods and concepts used by all researchers. Results could be subject to bias if methods and concepts are misunderstood. We mitigated this threat by establishing and discussing a review protocol to achieve a sufficient common understanding. We formulated inclusion and exclusion criteria to reduce bias due to subjective decisions in our selection process and conducted the data analysis independently. When opinions differed, the results were discussed until consensus was reached among the researchers. User Study. Although 61 participants provide a substantial sample size, some of the conclusions might be affected by it and should not be overgeneralized. Our strategy to select the participants has some limitations and might not reflect the whole population which may threaten the global generalizability of our results. The majority of our participants had profound information technology (IT) knowledge, i.e., this fact may not consider people who have difficulties operating software systems. However, we did not find any evidence of this threat impacting our results. Instead, we gained valuable insights into what different people think and what their attitudes are toward privacy. The perception and understandability of explanations is hard to measure. We handled this threat by using statistical tests where appropriate and using more qualitative analyses otherwise.

VII. CONCLUSION AND FUTURE WORK

Privacy is one of the human values in software engineering and is becoming increasingly important in our highly interconnected world. In order to provide end users with their right to (online) privacy when using software systems, it is important to enter into a dialogue with them about a system’s data practices. To this end, we researched a novel concept of user-centered privacy explanations, to educate end users regarding their privacy. Based on survey data and validated through an LR, we elicited high-level requirements for privacy explanations. We refined the requirements into context, content and consent (3C-principle) which shaped our design concept for user-centered privacy explanations. We evaluated our concept in a user study, by embedding it into a high-fidelity prototype. Our results suggest that our layered design concept, in line with the identified 3C-principle, enabled participants to understand the privacy aspects they regarded as important. In conclusion, we hold that our approach seems to be an appropriate way to communicate complex privacy information to end users and brighten their sensitivity with respect to their privacy awareness. As future work, we plan to integrate privacy explanations into an existing software system, as a next step to evaluate them in terms of suitability and usability.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122, Project ID 390833453), and supported under Grant No.: 470146331, project softXplain (2022-2025).

REFERENCES

Special Session TSECQS: Theoretical Software Engineering, Classical and Quantum Software Similarities
Quantum Software Models: Quantum Modules Tomography and Recovery Theorem

Iaakov Exman
School of Computer Science, Faculty of Sciences
HIT – Holon Institute of Technology
Holon, Israel
iaakov@hit.ac.il

Ariel Zvulunov
Software Engineering
The Jerusalem College of Engineering, Azrieli
Jerusalem, Israel
ariel.zvulun@gmail.com

Abstract—Quantum Tomography partially measures and then recovers the remaining density matrix quantum state, in order to verify that a certain device – processor or detector – indeed outputs the intended quantum state. However, single matrix element measurements rapidly increase in number with the density matrix dimension and are error-prone. This work proposes a novel quantum software viewpoint on quantum tomography. Instead of individual matrix elements, measurements and density matrix recovery are performed on higher-abstraction modules, i.e. semantically meaningful groups of matrix elements.

Quantum Modules Tomography potentially reduce the necessary number of explicit measurements, while still allowing recovery of the whole density matrix. A Recovery Theorem is formulated and proved: density matrix diagonal measurements suffice to recover the whole system density matrix, in terms of modules. The recovery procedure is illustrated by a few case studies.

Keywords—quantum software; quantum modules tomography; density matrix; software modules; density matrix recovery theorem

I. INTRODUCTION

Quantum Software Models deal with quantum software systems represented by density matrices. In a previous paper of this series [7], it has been shown that software system modules span sub-spaces, obtained by projectors acting on the whole software system state.

Quantum Tomography (QT) is a set of techniques to check behavior correctness of quantum devices, i.e. whether they output the intended quantum state, represented by a density matrix. QT measures some matrix elements, recovering the remaining density matrix elements, from the measured ones.

This paper argues that quantum modules tomography may reduce the number of necessarily measured matrix elements, and enables recovery of the whole density matrix of quantum software systems, from the matrix diagonal.

A. Quantum Software is Measurable

Quantum software is represented by a Density Matrix, whose most important characteristic is to be modularizable, as described in the next section. Quantum Software is a generic concept, that may refer to any of three types – quantum systems processing qubits, classical systems processing classical bits, or hybrid systems transitioning back and forth between quantum and classical sub-systems, due to the representation similarity of these three kinds of systems. Quantum software systems will be shown to comply with quantum computing measurements (see e.g. Nielsen and Chuang [12]). They are measurable, which is essential for Quantum Tomography.

B. Quantum Software Modules Tomography

There are two kinds of Quantum Tomography: quantum state tomography (e.g. Gross et al. [10]) reconstructing quantum states, and quantum process tomography which reconstructs processes by physical systems ([4],[13]).

Quantum Modules Tomography, proposed in this paper, is a novel kind of Quantum Tomography. Measurements and recovery of the whole software system density matrix are based upon system modules, instead of individual matrix elements. This is enabled by quantum software algebraic constraints.

C. Paper Organization

The rest of the paper is organized as follows. Section II characterizes Quantum Software. Section III deals with quantum software measurement. Section IV details Quantum Modules Tomography, then formulates and proves the Recovery Theorem for the whole density matrix. Section V illustrates the procedures with case studies and a simulation. Section VI concisely refers to related work. Section VII is a Conclusion of the paper.

II. QUANTUM SOFTWARE MODELS

This section introduces the reader to the main characteristics of Quantum Software Systems. It explains Density Matrix generation and modularization for these systems.

A. Conceptual Semantics and Bipartite Graph

Quantum Software systems display two main properties, structure and behavior, fitting two kinds of entities: Structors & Functionals. Structors generalize classes in Object Oriented Design (OOD). Functionals generalize OOD class methods.

DOI Reference Number: 10.18293/SEKE2023-214
Structors & Functionals have a double role. They are the concepts conveying meaning of the Quantum Software system. In the other role, their associated indices stand for vertices of graphs, which generate density matrices – the underlying linear algebraic representation of quantum software theory.

The Command Design Pattern ([9], page 233) is a running example in this paper, to explain basic ideas. Its goal is to abstract typical commands – copy, paste, delete, save – found in a variety of useful applications, into a reusable generic pattern. Behind every Structor \( S_j \) or Functional \( F_k \) indices there is a real meaningful concept understandable by software engineers.

The Command Design Pattern conceptual semantics is collected in Fig. 1. The Command module contains the command abstraction. Its Structors are ICommand, a generic interface, and Concrete Command standing for any commonly used command. Their Functionals enable to specify or execute commands. The Invoker module stands for menu-items or buttons used to invoke commands. The Receiver module refers to either a file or a formatted document receiving the outcome of a command execution.

A simple depiction of a quantum software system is a bipartite graph [15]. These graphs have two vertex sets, where a vertex is linked only to vertices in the other set. One set has Structor vertices \( \{ S_1, S_2, ..., S_j \} \) and another set has Functional vertices \( \{ F_1, F_2, ..., F_k \} \). A Structor providing a Functional – say, a class providing a method definition, in OOD parlance – is linked to the Functional.

The Command Design Pattern bipartite graph is seen in Fig. 2. Algebraic manipulations for modularization of quantum software systems, are totally independent of the specific conceptual semantics of the system indices.

B. From Bipartite Graph to Density Matrix

A Laplacian matrix \( L \) [11], [18] is defined upon any graph, in particular a bipartite graph, according to equation (1).

\[
L = D - A
\]

where \( D \) is the diagonal Degree matrix – showing the degree \( d_{jj} \) of vertex \( j \) – and \( A \) is the Adjacency matrix – showing the neighbors of each vertex with a 1-valued matrix element, with negative sign due to equation (1).

It has been observed by Braunstein et al. [3] that a Density Matrix \( \rho \) can be obtained from a Laplacian \( L \) normalized by its Trace, the sum of the diagonal degrees, as in eq. (2).

\[
\rho = L / \text{Trace}(L)
\]

A Density matrix \( \rho \) generated from the bipartite graph in Fig. 2, by equation (1) and normalized as in eq. (2) can be seen in Fig. 3.

C. Modules from the Density Matrix

Modules of Quantum Software systems can be obtained directly from the Laplacian eigenvectors fitting zero-valued eigenvalues (see [6]). Modules can also be obtained by partition of the sum of projection operators fitting kets of the system basis set, into disjoint sets of projection operators, as explained in [7]. Modularized matrices of a Software system have mutually orthogonal modules.

Figure 1. Command Design Pattern Modules, Structors & Functionals – It has 3 modules, Command, Invoker, Receiver, with sizes 2-by-2, 1-by-1, 2-by-2. For instance, the Command Module Structors & Functionals set is \( \{ S_1, S_2, F_1, F_2 \} \). (Figures in color online).

Figure 2. Command Design Pattern Bipartite Graph – Its 10 vertices are: 5 (green) Structors \( \{ S_1, S_2, ..., S_5 \} \) and 5 (orange) Functionals \( \{ F_1, F_2, ..., F_5 \} \). Also seen 3 (blue) background Modules. The middle module has \( \{ S_3, F_3 \} \) as vertices. Arrows link Structors to provided Functionals, e.g. S2 provides F1 & F2.

Figure 3. Command Design Pattern Density Matrix – One can see the Laplacian matrix \( L \) within square brackets, normalized by the Trace = 14. Within the Laplacian, one perceives the Diagonal Degree matrix \( D \) (green) and the two quadrants of the Adjacency matrix \( A \) (blue), above the diagonal and reflected below the diagonal. Above the column indices one sees kets, and to the left of the row indices the bras of this software system basis set.

Algebraic features of software systems represented by a Density Matrix derived from the Laplacian are essential for this paper’s goal, viz. to facilitate recovery of the quantum software system state. One can easily show, from eq. (1) that each of the Laplacian rows and columns sum to zero. This is preserved in the Density Matrix – and perceived e.g. in Fig. 3.

The Adjacency matrix of a Quantum Software system, also appearing in eq. (1), has two quadrants, as seen in Fig. 3. The upper-right quadrant is above the diagonal and reflected into the lower-left quadrant below the diagonal. Further algebraic properties of relevance to Quantum Module Tomography are collected in Definition 1, in sub-section IV.B.
III. QUANTUM SOFTWARE MEASUREMENT

This section shows that Quantum Software systems comply with quantum computing projective measurements [12] for modules, illustrated with the running example software system.

A. Quantum Software Projective Measurement

Module projectors span sub-spaces of a whole Quantum Software system. A natural choice for modules is projective measurement, defined by its observables and final state.

An observable is a Hermitian operator $M_m$ on the state space of the whole system Density Matrix $\rho$. Its properties are:

- mutually orthogonal observables and spectral decomposition as in eq. (3).

$$M_m = \sum_n n \cdot P_n$$  \hspace{1cm} (3)

$P_n$ is a projector onto the $M_m$ eigenspace with eigenvalue $n$.

The final state of the system $S_m$ after measurement is obtained by equation (4).

$$S_m = (M_m \rho M_m^\dagger)/\text{Trace}(M_m^\dagger M_m \rho)$$  \hspace{1cm} (4)

where $M_m^\dagger$ is the conjugate transpose of $M_m$.

B. Quantum Software Module Observables

The upper module $\{S_1, S_2, F_1, F_2\}$ in each quadrant of Fig. 3 exemplifies eq. (3) by its observable as a projectors’ sum:

$$M_1 = |0000\rangle \langle 0000| + |0001\rangle \langle 0001| + |0101\rangle \langle 0101| + |0110\rangle \langle 0110|$$  \hspace{1cm} (5)

This observable is seen as a full-matrix in Fig. 4.

Figure 4. Observable Matrix of Upper Module of the Command Design Pattern in Fig. 3 – The only non-zero matrix elements in this observable are in the diagonal, fitting the eq. (5) projectors. Matrix element indices, kets and bras of this Module are seen. The module locations in $\rho$ are marked as blue background.

B. Quantum Software Module Measurement Final State

Now we insert the Observable $M_1$ into eq. (4) to exemplify the final state after measurement of the Command Design Pattern. Equation (4) is simplified, as $M_1$ is real, diagonal and has only 1-valued matrix elements. Also the normalizing factor $(1/14)$ of $\rho$ in Fig. 3 cancels out, as it appears in both numerator and denominator of eq. (4). The resulting final state is in Fig. 5.

This final state is an interesting result, since the whole system Density Matrix can be obtained from a bidirectional direct sum of the modules reduced density matrices (see [8]).

IV. QUANTUM MODULES TOMOGRAPHY

This section states the idea of Quantum Modules Tomography and presents its overall procedure. The Recovery Theorem enabling a more efficient Quantum Software Density Matrix recovery is formulated and proved.

A. Idea of Quantum Modules Tomography

Abstraction is the basic idea behind software modules and any of their applications. Abstraction means reducing the number of concepts necessary to formulate the purpose and functionality of a system and its sub-systems. The benefits are deeper understanding and increased efficiency.

Quantum Modules Tomography is based upon two assumptions:

- **Reduction of measurements number** – to at most proportional to the size of the diagonal Degree matrix;
- **Constraining of Density Matrix completion** – based upon the Laplacian standard algebraic features collected below in Definition 1, in the next sub-section.

An important observation is that the number and sizes of Modules in a software system is finite and small. This is justified by dealing with software systems in a hierarchical way. The same is true concerning the numerical values of the diagonal Degree matrix elements which express the relative dimensions of the modules.

B. Towards the Recovery Theorem: Density Matrix Algebraic Features

The standard algebraic features of the Laplacian are based upon the Laplacian standard algebraic features collected below in Definition 1, in the next text box.
C. The Recovery Theorem

After the needed measurements are done for a quantum software system, the next theorem assures that one can recover lacking values to complete the system density matrix.

**Theorem 1 – Recovery of whole Density Matrix from its Diagonal**

**Assuming:**

a- quantum software system – is describable by a modular density matrix \( r \) obtained by normalizing a Laplacian \( L \) as \( r = L / \text{Trace}(L) \), where \( L \) complies with the Standard Laplacian algebraic features in Definition 1;

b- strictly necessary measurements – were performed to obtain the Density Matrix diagonal;

**Then:**

The whole quantum software system Density Matrix is completely recoverable by a finite small set of additions to the off-diagonal density matrix elements.

**Proof:**

The number of the off-diagonal density matrix additions is obviously finite. It is enough to complete the set of additions to the upper-right quadrant of the Adjacency matrix. The lower-left quadrant is the reflection of the upper-right quadrant. The exact number of additions to the upper-right quadrant of the Adjacency matrix is half of the Trace of the whole matrix.

For each diagonal 1-valued matrix element one should add one negative 1-valued matrix element in the same row and/or column. For each diagonal 2-valued matrix element one should add two negative 1-valued matrix elements in the same row and/or column. And so on for higher valued diagonal matrix elements. All the remaining matrix elements above the whole matrix diagonal are zero-valued, and reflected below the diagonal.

The above unequivocally determines all the density matrix element numerical values to be added. The exact addition locations are constrained by the standard Laplacian algebraic features.

We emphasize an essential aspect of the above theorem:

- the number of additions is small as module sizes are limited by collecting only related concepts in each module, to facilitate human understanding.

D. Quantum Modules Tomography Procedure

The Quantum Modules Tomography Procedure is:

1) Measure whole matrix degree diagonal elements;
2) Fill the Adjacency matrix diagonal of the upper-right quadrant with negative 1-valued elements;
3) Add other non-zero elements in the Adjacency matrix upper-right quadrant according to the Density Matrix degree diagonal;
4) Complete with zero-valued elements above the Density Matrix degree diagonal;
5) Reflect the upper-right quadrant to elements below the Density Matrix degree diagonal.

V. CASE STUDIES

This section illustrates the Quantum Modules Tomography procedure by means of two Quantum Software systems: the Grover Search design, a strictly quantum system, and a simulation of a simplified classical system.

A. Quantum Grover Search

Grover Search (see e.g. [2] page 166) is a well-known quantum algorithm which performs search of unstructured (unsorted) \( N \) data items with a reduced complexity of \( O(\sqrt{N}) \), instead of the classical complexity of \( O(N) \).

Modules, Structors and Functionals of strictly quantum systems (processing only qubits) are obtained from quantum circuits, as in Fig. 6.

![Figure 6. Grover Search quantum circuit - Structors are "boxes" (green) containing quantum gates (orange), except for the typical “measurement” that inputs qubits and outputs classical bits. Modules (blue) are Structors grouped by some logical reason. For instance, the Grover Iteration is a single cycle of a loop consisting of an Oracle and an Amplification Structor.](image)

The Grover Search Modules, Structors & Functionals are shown in Fig. 7.
Figure 7. Grover Search Modules, Structors & Functionals – It has the same color conventions as in Fig. 1. $H$ stands for the Hadamard operator, here to the $n^{th}$ tensor product power.

Once one has the Structors & Functionals for the quantum software system – be it purely classical, strictly quantum, or hybrid – from a bipartite graph one generates and normalizes the Laplacian to a Density Matrix.

The Grover search quantum software system Density Matrix is in Fig. 8. It differs from purely classical systems, as the Command Pattern, by the number of basis set qubits needed to describe the system, and the module sizes.

Figure 8. Grover Search Density Matrix – This software system Density Matrix has a 3 qubits basis set and 3 modules, only one with a 2-by-2 size.

The Grover Search case study illustrates that:

- The same algebraic techniques are applicable to all quantum software systems – pure classical, strictly quantum or hybrid (classical and quantum).
- Ignoring Structors & Functionals location and semantics, a quantum Grover Search 2-by-2 module in Fig. 8 or a classical Command design pattern 2-by-2 module in Fig. 3 are identical and measurable by an observable comparable to that in Fig. 4.

B. Simulation of a Simple Classical System

The purpose of this simulation is to illustrate the idea that Quantum Module Tomography indeed may recover the whole Density Matrix of a Quantum Software system, while reducing the number of needed matrix element measurements. The simulation was done on the simplified Software System, whose density matrix is seen in Fig. 9.

Figure 9. Simplified Quantum Software System Density Matrix – it is obtained from Fig. 8 rows/columns $\{F2,F3,F4\}$ and $\{S2,S3,S4\}$ suitably renumbered. It uses the same color conventions as in previous figures. It has two modules.

The measurements projectors are in Fig. 10 with their probabilities.

The overall simulation – implemented in qiskit, an IBM quantum computing oriented language – performed “projectors” sampling upon the Density Matrix in Fig. 9, using the $\text{DensityMatrix.sample_memory}$ function. Once the sampling achieved a stable probability distribution, one obtains the values of each of the diagonal matrix elements, as specified in the Quantum Modules Tomography Procedure in sub-section IV.D. These values confirmed the expectations of Fig. 10.

When does the measured probability distribution reach stability? The answer is given in Fig. 11.

Figure 11. Convergence of sampling percentages – one perceives that by increasing measurement numbers, graphs converge to two values 12% and 24% already for 200 Measurements and is stable for 250 Measurements. When samplings are stable, one may stop the simulation. These percentages respectively fit the 1/8 and ¼ probabilities.
For instance, looking at the 250 measurements, since the heights of the 12% results are twice those of 24% results, then the total value for 12% is \(12\times 0.5 = 6\) and the total value for 24% is \(24\times 0.5 = 12\). This fits the facts that there are 4 projectors with probability 1/8, i.e. \(4\times (1/8) = \frac{1}{2}\) and only two projectors with probability \(\frac{1}{4}\), i.e. \(2\times (1/4) = \frac{1}{2}\). In other words, the simulation results confirm the expectations of Fig. 10.

VI. RELATED WORK

Here are very concise references to topics relevant to this paper.

Braunstein and co-authors [1] focus on mixed states separability, represented by density matrices obtained from Laplacians. They state that specific graph types originate entangled or separable states independently of their labeling. Chai Wah Wu [19], [20], also considers Laplacian and density matrix separability.

The Quantum Tomography Measurements literature is very extensive. We provide a few entry points. The “Quantum State Estimation” book edited by Paris and Rehacek [13] has chapters relevant to Tomography. For instance, the D’Ariano et al. [14] chapter on “Quantum Tomographic Methods”, and the chapter by Altpeter et al. on “Quantum State Tomography” [1].

The maximum-likelihood (MaxLik) algorithm, and its variants, e.g. the Diluted Maximum-Likelihood algorithm by Rehacek et al. [14] offer alternative ways to reconstruct a quantum state from tomographic measurements.

VII. CONCLUSION

A. Theory and Potential Applications

The potential applications refer to eventually reducing the number of measurements, and faster recovery of quantum software systems’ density matrix, according to the Recovery Theorem enabling density matrix reconstruction from measurement of the density matrix diagonal only.

Two caveats are in place here. This paper theory focused on density matrices derived from Laplacians. Elsewhere, we discussed transformations of any density matrices to quantum software density matrices as derived from Laplacians.

Second, the ultimate testing of the ideas of the “Quantum Modules Tomography” theory will be in actual tomography measurements in a laboratory. This is an invitation for experimental researchers.

B. Future Work

Open issues to be dealt with in future work include different formulations of Recovery Theorems, solving particularly difficult modularizations of quantum software systems, and concurrent use of alternative reconstruction density matrix algorithms, such as maximum-likelihood, together with the Recovery Theorem of this paper.

C. Main Contribution

The main contribution of this paper is the Modules abstraction for Quantum Software Systems, resulting in the Recovery Theorem, with potential applications to Quantum Modules Tomography.

REFERENCES

[7] Eric Gamma, Richard Helm, Ralph Johnson and John Vlissides. 1995. Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston, MA, USA.
Automatic Diagnosis of Quantum Software Bug-Fix Motifs

Krishn V. Kher¹, M. Bharat Chandra¹, Ishan Joshi², Lei Zhang², and M. V. Panduranga Rao¹

¹Department of Computer Science and Engineering, IIT Hyderabad, India
²Department of Information Systems, University of Maryland, Baltimore County, USA

Abstract

Bug-fix pattern detection has been investigated in the past in the context of classical software. However, while quantum software is developing rapidly, the literature is still lacking automated methods and tools to identify, analyze, and detect bug-fix patterns. To the best of our knowledge, our work is the first to leverage classical techniques to detect bug-fix patterns in quantum code. In this paper, we propose an automated framework, called Q-Diff, for detecting bug-fix patterns in IBM Qiskit quantum code. In the framework, we develop a proof-of-concept tool based on Abstract Syntax Trees. To validate our method, we test Q-Diff with a variety of quantum bug-fix patterns using examples. We hope our work will attract the attention of the quantum software engineering community to improve the quality of quantum software.

1 Introduction

With the increasing size and complexity of quantum programs being written, it is natural to expect an increased number of bugs and more complicated bugs to creep into quantum source code. Indeed, this phenomenon is folklore in classical software [3]. Therefore, a significant body of research, tools, and techniques exist in the detection and elimination of bugs in classical software [8]. These techniques range from static code analysis [2] to run-time detection [28].

The effort to understand and classify commonly occurring bugs yields rich dividends. Steps for this approach include the identification and classification of bug patterns, the design of bug-fixes, and the detection of bug-fix patterns. Correct identification of bug-fix patterns is of immense use in statistical analysis of bugs, their prevalence, and fixes. This helps in streamlining and developing tools for automatic bug detection, fixing, and manpower training.

In this paper, we are particularly interested in automated approaches to detect bug-fix patterns in quantum programs.

In classical software engineering, common bug-fix patterns are well studied, and both manual and automated approaches have been proposed [4, 12, 18, 20]. In quantum software engineering [19], preliminary studies exist in testing of quantum programs [14, 15, 16], and bug patterns of quantum programs [6, 29, 30].

In this paper, we propose a new framework, called Q-Diff, for detecting quantum bug-fix patterns. We translate the framework into a tool, as a proof-of-concept. The tool uses Abstract Syntax Trees (AST) to compare the quantum buggy code and patches by extracting required information specific to bug-fix patterns (see Section 3). The detectors in the tool then positively identify or reject the bug-fix pattern. We validate this tool with three quantum bug-fix patterns in Qiskit [23] (in Section 4). In accordance with the framework, this in-house tool will be further developed, incorporating more quantum bug-fix patterns and supporting more quantum programming frameworks.

Our contributions are: 1) we develop an AST-based tool to detect bug-fix patterns in Qiskit,¹ which is, to the best of our knowledge, the first work in the literature, and 2) we show that our tool can detect at least three bug-fix patterns.

2 Related Work

In the interest of space, we assume a working knowledge of quantum computing [17]. We provide a brief review of existing bugs, fixes, and bug-fix patterns in both classical and quantum programs.

2.1 Classical Bug and Fix Patterns

Bug-fix patterns for classical programs are widely studied in the literature. Pan et al. [18] identify 27 bug-fix patterns based on an analysis of historical bug-fix pairs. Cam-

¹Our source code is publicly available at https://github.com/KrishnKher/Q-AutoDiaBFM.

Automated approaches for detecting certain bug-fix patterns are considered more efficient compared to manual approaches in general. Madeiral et al. [12] manually analyze hundreds of bug-fixes and propose an automated tool based on AST to detect repair patterns in bug-fixes using the GumTree algorithm. The AST method is an effective solution in terms of detecting code differences, other AST-based auto detection tools include [7, 13].

2.2 Quantum Bug and Fix Patterns

The literature on testing and debugging quantum programs is growing. A quantum program is challenging to test because of the underlying principles of quantum mechanics [14]. Software engineering principles are being applied to quantum program testing and debugging [14, 15]; see [29] for a comprehensive overview of quantum software engineering research work.

The community takes multiple approaches to tackle the challenge. Testing quantum programs may be simplified by adding assertion checks to the code [1, 6, 9, 10] or, in some cases, introducing debugging tricks, such as extracting classical information [15]. We can also adapt classical fuzzy testing techniques [24] or perform property-based testing [5]. The identification of bug patterns in quantum programs can assist in defect analysis and categorization [11, 30]. Zhao et al. [31] propose a benchmark to evaluate testing and debugging methods for Qiskit programs. As the research of quantum software engineering is still in its infancy, the literature lacks automated solutions to detect bug-fix patterns in quantum programs.

3 Our Approach

Code-diffing is a technique used to compare two versions of code to identify differences or changes. This is typically done by comparing the ASTs of the buggy and the fixed code, line-by-line, and identifying added, deleted, or modified lines. Figure 1 shows the architecture of our tool Q-Diff. As can be seen, Q-Diff reads both buggy and fixed code, then determines the bug-fix pattern based on a sequence of steps, including 1) obtaining the AST, which abstracts the structured code information, 2) matching against syntactic checks using regular expression (RegEx) formulas, which searches for the quantum-related syntax, and 3) performing semantic checks, to identify bug-fix patterns based on quantum code semantics.

![Figure 1: The architecture of Q-Diff.](image)

As an initial study, Q-Diff can detect three bug-fix patterns, i.e., incorrect initialization of qubits, incorrect gate operations, and incorrect measurements, as can be seen in Figure 1. Note that each bug-fix pattern has its own detector, and each detector has its own logic that determines whether a bug-fix code pair falls into this category or not. In other words, three detectors, corresponding to the three bug-fix patterns, have been implemented in Q-Diff at present. However, the sequence of logic in each detector is the same, as depicted in Figure 1. This code structure can be extended to add more detectors and to change or augment the logic of various detectors if necessary.

There are two advantages of the current implementation of multiple detectors. First, each detector can run in parallel, in principle. If a large number of bug-fix code pairs are waiting to be fed into Q-Diff, we do not need to input each pair sequentially. Second, there may exist hybrid bug-fix patterns where a bug-fix pair can be categorized into multiple patterns. In such a case, the detectors will not interfere with each other. For future work, we propose to adopt a hierarchical approach. Starting with a highly coarse-grained categorization like Python-related bug-fix patterns and Quantum-related bug-fix patterns in Qiskit, we descend to finer grains of categorization, ending in detectors for specific bug-fix patterns at the “leaf” level. We believe that this approach has the potential to improve search efficiency substantially whenever large categories can be pruned out.

We now discuss how our framework operates in detail.

3.1 Creation of Bug-Fix Pattern List

We identify bug-fix patterns from various sources, including the studies of [11, 31], StackOverflow, and GitHub.
repositories.\(^2\) Based on our initial study, we create representative examples of commonly occurring buggy and fixed code for each of the bug-fix patterns. These examples are derived from real-world examples. We then patch them up manually and pass the code pair to \textit{Q-Diff}. For testing the detector of each bug-fix pattern, we create a group of test cases with an equal number of positive test cases (matching the pattern) and negative test cases (not matching the pattern). The detectors should correctly identify the positive cases and reject the negative cases.

### 3.2 AST Extraction

We first extract the ASTs for the buggy and fixed code, respectively. These ASTs will subsequently provide information to the detectors for syntactic and semantic analysis. Example information that ASTs provide includes identifiers and a number of quantum circuit objects.

### 3.3 Bug-Fix Pattern Detectors

The implementation of a detector varies from one bug-fix pattern to another. However, they all have an initial syntactic filter and a semantic check module.

#### 3.3.1 Syntactic Checks with RegEx Formulations

For each bug-fix pattern, we formulate a RegEx. The RegEx is used to identify the lines of code that are relevant to a particular bug-fix pattern. If a match is found on a line of code, then we move on to the semantic check phase. When all lines of code are exhausted without finding a match, we declare that the buggy-fix code pair under investigation does not belong to the current pattern.

#### 3.3.2 Semantic Checks

After RegEx matching, we perform semantic checks. These checks would be specific to the bug-fix pattern detector. For example, in the context of incorrect gates, we are still not sure if a line of code matched by the RegEx contains an object of quantum circuits. Thus, we analyze the information extracted from the AST in the context of Qiskit semantics to decide this.

### 4 Implementation of \textit{Q-Diff}

In this section, we discuss examples of bug-fix patterns and their detection. Note that the examples described here are selected to illustrate the working of \textit{Q-diff}. We provide more examples on our GitHub repository.

As discussed in Section 3, the bug-fix patterns that we describe here are 1) incorrect initialization, 2) incorrect gates, and 3) incorrect measurements. These three patterns correspond to the three key elements in quantum computation, i.e., qubits, operations, and measurements. We emphasize that we only consider single-line errors instead of multi-line errors. For most simple code of this nature, multi-line errors do not happen. However, \textit{Q-Diff} can be extended in the future to identify multiple independent single-line bug-fix patterns in a buggy-fixed code pair. We now look at detectors for the three bug-fix patterns with examples.

#### 4.1 Incorrect Initialization

\textit{Q-Diff} detects two cases of incorrect initialization: 1) A gate operation is applied on a wrong qubit, i.e., it should be applied on a different qubit, as an IncorrectInitialization error; 2) The number of qubits that the QuantumCircuit works with is different in the buggy and fixed code—the so-called IncorrectQubitCount error.

##### 4.1.1 Syntactic Checks

We use RegEx \texttt{.+..*} and \texttt{.+QuantumCircuit.*} to retrieve the lines of the code where a Qiskit gate has been used.

##### 4.1.2 Semantic Checks

We conduct a two-step semantic check. First, we identify the lines where the same kind of gate is being used, to separate a case from a IncorrectGate error (here, quantum gates are used for qubit initialization). Next, we check each of these valid gates if there is any difference in the qubit indices in the bug-fix code pair. If there is any difference then we flag it as an IncorrectInitialization error.

#### 4.1.3 Examples

Listing 1: Incorrect initialization

Listing 1 simulates a scenario where an incorrect qubit is initialized—the developer wants to initialize the second qubit instead of the first one, which is semantically different, although syntactically correct. The only error here is...
in the argument to the Hadamard gate being applied. Our semantic checks identify the bug-fix and correctly classify it as an IncorrectInitialization error.

4.1.4 IncorrectQubitCount

The case for IncorrectQubitCount is very similar—there is a mismatch in the argument for the QuantumCircuit object, i.e., in the number of qubits with which the QuantumCircuit object is instantiated. This is dealt with in the same way as above; thus, we skip the details.

4.2 Incorrect Gates

Here, we show instances where an incorrect gate is applied on a certain specified qubit of a QuantumCircuit.

4.2.1 Syntactic Checks

The RegEx for this type of bug is \[.+\] which is the same as the first one that we use for IncorrectInit. However, the semantic check will be different, and we will explain the details in Section 4.2.2. The RegEx abstracts lines of code involving a quantum gate. After identifying the gate operations, we will identify the name of the quantum circuits and gates for comparison.

4.2.2 Semantic Checks

After the syntax checks from RegEx, we perform additional semantic checks in two steps. First, we check 1) if the identifiers are not equal, and 2) if they both actually belong to the inbuilt gates available in Qiskit. If either of these checks fails, we declare the bug-fix pair is not of an IncorrectGate type. If both conditions are satisfied, Q-Diff classifies the bug-fix pair in this category (as shown in Listing 2). Next, we identify QuantumCircuit objects independent of their actual name in the code and detect errors, using the AST. Listing 3 illustrates this case.

4.2.3 Examples

Listing 2 (derived from Stack Overflow [22]) illustrates a scenario where the gate operation is incorrect in QuantumCircuit. The identifiers are \(h\) and \(x\) in this example, both of which correspond to valid inbuilt gates in Qiskit, namely, the Hadamard gate and the X-gate. Obviously, the identifier names, \(h\) and \(x\) are not equal. Hence, it is classified as an IncorrectGate kind of bug.

Listing 3 is very similar to the first example, except that now the QuantumCircuit object has different names, i.e., \(a\) and \(qc\). In the second example, we first check if the underlying gate is being accessed by a QuantumCircuit object. We do this using the data extracted from the AST of the buggy and the fixed code. This removes the dependency on the identifier name to determine if a bug-fix pair is in the IncorrectGate or not.

4.3 Incorrect Measurements

Here, we describe the bug-fix pattern of incorrect measurements of qubits, i.e., IncorrectMeasurement in Q-Diff.

4.3.1 Syntactic Checks

We use a different RegEx expression compared with the first two patterns, i.e., \[.+\].measure.* This RegEx helps us to retrieve the code where a Qiskit measure() function is used for further semantic checks.

4.3.2 Semantic Checks

We categorize a bug-fix pair in this class as follows.

First, we check if there are any measure functions in the bug-fix pair. If not, we declare that the bug is not of an IncorrectMeasurement type; otherwise, Q-Diff goes to the next step.

Second, there are three commonly used measure functions in Qiskit, namely, measure(), measure_all(), and measure_inactive(). The Q-Diff first computes the numbers of all variants of measure functions used in the buggy and fixed code, respectively. If the numbers are different, the bug-fix pair then falls into this category; otherwise, Q-Diff will go to the next step. Note that we assume multiple measure functions are in one-to-one correspondence in the buggy and fixed code.
Third, if `measure` functions are different in the buggy and fixed code, this bug-fix pair falls into this category; otherwise, `Q-Diff` will go to the next step.

Fourth, if the `measure` functions are the same, we check if the arguments passed to the `measure` functions are the same in a bug-fix code pair. Since arguments are usually qubit lists, we use Python `numpy` arrays to check the difference. If the arguments are different, then it is in the category of IncorrectMeasurement; otherwise, `Q-Diff` will go to the last step of the semantic check.

Finally, if both the `measure` functions and the arguments are the same, we check the positions of the measurements. If the positions are different in the buggy and fixed codes, this code pair also falls into the IncorrectMeasurement category.

### 4.3.3 Examples

#### Listing 4: First example of incorrect measurement

We show two examples here. In Listing 4 (derived from `qiskit-terra` issue report #6751), though valid `measure` functions are used in both bug and fix, `measure_all()` function shows in “bug” while `measure()` function is its replacement in “fix”. The outputs of the `measure_all()` function and the `measure()` function are not expected to be the same, hence an IncorrectMeasurement error.

```python
Buggy Code:
1 qr = QuantumRegister(2, name='qreg')
2 cr = ClassicalRegister(2, name='creg')
3 qc = QuantumCircuit(qr, cr)
4 qc.h(qr)
5 - qc.measure_all()
Fixed Code:
6 qr = QuantumRegister(2, name='qreg')
7 cr = ClassicalRegister(2, name='creg')
8 qc = QuantumCircuit(qr, cr)
9 qc.h(qr)
10 + qc.measure(qc.qubits, qc.clbits)
```

#### Listing 5: Second example of incorrect measurement

Similar to Listing 4, although valid `measure` functions are being used in Listing 5 (issue report #664 in qiskit-aer), the outputs written to the classical bits are in a wrong order in the buggy code. Hence, this is an IncorrectMeasurement error, and `Q-Diff` identifies it successfully.

```python
Buggy Code:
1 qr = QuantumRegister(2)
2 + qc.h(0+1)
3 ... 
Fixed Code:
4 qc = QuantumCircuit(2)
5 + qc.h(1)
6 ... 
```

#### Listing 6: Code identified as a correct implementation

In Listing 6, we notice that the underlying logic in both the code is exactly the same, but for the representation, wherein the buggy code, qubit 0 is denoted by qubit 0 + 1. A manual check determines this is not a bug-fix pattern and our tool does the same—it is capable of recognizing the semantics as opposed to just reporting text differences.

### 4.4 A “Counter Example” Epilogue

How would `Q-Diff` behave if there is a syntactic (or textual) difference between the buggy and the fixed code, but the difference has no semantic consequence?

```python
Buggy Code:
1 qc = QuantumCircuit(2)
2 + qc.h(0+1)
3 ... 
Fixed Code:
4 qc = QuantumCircuit(2)
5 + qc.h(1)
6 ... 
```

#### Listing 6: Code identified as a correct implementation

In this paper, we conduct the first analysis of bug-fix patterns in quantum programs. We propose an AST-based framework called `Q-Diff` to automatically detect Qiskit bug-fix patterns and prove the concept with various examples. An obvious future direction is to enhance the tool with detectors for an exhaustive list of bug-fix patterns and prove the concept with various examples. We also hope the quantum software engineering community will expand methods and tools to improve the quality of quantum software.

### 5 Threats to Validity

Validity threats are classified according to [26, 27]. Internal and construct validity: In this initial study, we test `Q-Diff` with bug-fix patterns where only one line of code is modified. We develop both positive and negative test cases to verify our implementation as proof-of-concept and will test our framework with real bug-fix code. External and conclusion validity: Software engineering studies suffer from the generalization problem, which can only be solved partially [25]. Although we cover the three most significant bug-fix patterns of Qiskit in quantum computations, our findings may not generalize to other projects. As a pilot study of quantum bug-fix patterns, we plan to extend our research with more bug-fix patterns and quantum frameworks. We also hope the quantum software engineering community will expand methods and tools to improve the quality of quantum software.

### 6 Conclusions and Future Work

In this paper, we conduct the first analysis of bug-fix patterns in quantum programs. We propose an AST-based framework called `Q-Diff` to automatically detect Qiskit bug-fix patterns and prove the concept with various examples. An obvious future direction is to enhance the tool with detectors for an exhaustive list of bug-fix patterns. Other bug-fix patterns that can be potentially detected by `Q-Diff` include but are not limited to 1) incorrect definition or application of custom gates in Qiskit and 2) unhandled exceptions in Qiskit. A second direction to pursue is the detection of more complex bugs and fixes, especially composite ones, which involve multiple bug-fix patterns and fixes.
that involve multiple lines of code that are not necessarily co-located in the code. An example is the “computation in the wrong basis” bug-fix pattern. Several quantum algorithms first change the basis (say, from computational to Fourier), apply unitary gates $U_i$ and measurements $M_i$, and revert (to computational basis) for further computation—the buggy code could then be $U_1 M_1 \ldots$ and the fixed code is $H U_1 M_1 \ldots H^{-1}$, where $H$ is the Fourier transform. Finally, a multi-pronged approach that uses ASTs, parse trees, and measurements is revert (to computational basis) for further computation—

References


Symbolic Model Checking Quantum Circuits in Maude

Canh Minh Do and Kazuhiro Ogata
School of Information Science
Japan Advanced Institute of Science and Technology (JAIST)
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
Email: {canhdo,ogata}@jaist.ac.jp

Abstract—This paper presents a symbolic approach to model checking quantum circuits by using a set of laws from quantum mechanics and basic matrix operations with Dirac notation. We use Maude, a high-level specification/programming language based on rewriting logic, to implement our symbolic approach. As a case study, we use the approach to formally specify and verify the correctness of the quantum teleportation protocol, which is an important quantum communication protocol in the early work of quantum communications. Moreover, our implementation can be used as a general framework to formally specify and verify quantum circuits in Maude in an effortless way, where only an initial quantum state and a sequence of actions describing how a quantum circuit works in a simple way are required.

Keywords—quantum circuits; Dirac notation; symbolic model checking; Maude

I. INTRODUCTION

Quantum circuits are a model of quantum computation used in quantum computing. They are composed of a sequence of quantum gates, measurements, initializations of qubits, and possibly other actions. Quantum gates operate on quantum bits (qubits), the quantum counterpart of classical bits, and manipulate the state of a quantum system to perform quantum computations. The outputs of quantum circuits are quantum states, which can be measured to obtain classical outcomes with probabilities from which other actions can take place. Quantum circuits play a crucial role in quantum algorithms as they are used to design and implement quantum algorithms before actually running on quantum computers. Because quantum computing is counterintuitive and radically different from classical computing, the likelihood of errors in quantum algorithms and circuits is much higher than in classical algorithms. Therefore, it is critical to verify that quantum circuits (or algorithms) enjoy desired properties. There is a symbolic approach [1] to (semi-)automatically reasoning about quantum circuits in Coq\(^1\), an interactive theorem prover, but it often requires human users to provide necessary lemmas to complete its proofs. Meanwhile, model checking is a formal verification technique widely used in both academia and industry to automatically verify that a system satisfies some desired properties. Although there are some model checkers dedicated to quantum programs [2], [3], there is still a gap between model checking quantum programs and quantum circuits, which should be filled in.

In this present paper, we propose a symbolic approach to model checking quantum circuits by using a set of laws from quantum mechanics and basic matrix operations with Dirac notation [4]. Concretely, quantum states, quantum gates, and measurements are described in Dirac notation instead of using explicitly complex vectors and matrices as Paykin et al. proposed in [5], making our representations more compact. Using the set of laws, we can automatically reason about quantum operations on quantum data, such as qubits. We use Maude [6], a high-level specification/programming language based on rewriting logic, to formalize quantum states, some basic gates (e.g., Hadamard and controlled-NOT gates), and measurements on a standard basis with Dirac notation. As a case study, we use our approach to analyze the quantum teleportation protocol, which is an important quantum communication protocol in the early work of quantum communications. Moreover, our formalization takes the probabilities into account and so we are able to analyze both the quantitative and qualitative properties of the quantum teleportation protocol with a built-in LTL model checker in Maude. Although we only use the quantum teleportation protocol as a case study, our implementation can be used as a general framework to formally specify and verify quantum circuits in an effortless way, where only an initial quantum state and a sequence of actions describing how a quantum circuit works in a simple way are required. Our implementation is publicly available at https://github.com/canhminhdo/QTC-Maude.

The rest of the paper is organized as follows: §II Preliminaries, §III Symbolic Reasoning, §IV Quantum Teleportation Protocol, §V Formal Specification, §VI Symbolic Model Checking, §VII Related Work, and §VIII Conclusion.

II. PRELIMINARIES

This section briefly describes some basic notations from quantum mechanics based on linear algebra and Kripke structures.

---

1https://coq.inria.fr/
A. Basic Quantum Mechanics

In classical computing, the fundamental unit of information is a bit whose value is either 0 or 1. In quantum computing, the counterpart is a quantum bit or qubit, which has two basis states, conventionally written in Dirac notation [4] as $|0\rangle$ and $|1\rangle$, corresponding to one-bit classical values, whose values are two column vectors $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, respectively.

In quantum theory, a general state of a quantum system is a superposition or linear combination of basis states. A single qubit has state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha$ and $\beta$ are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. States can be represented by column complex vectors as follows:

$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha|0\rangle + \beta|1\rangle$, where $\{|0\rangle, |1\rangle\}$ forms an orthonormal basis of the 2D complex vector space. The basis $\{|0\rangle, |1\rangle\}$ is also called as the standard basis. Formally, a quantum state is a unit vector in a Hilbert space $H$, which is equipped with an inner product satisfying some axioms.

The evolution of a closed quantum system can be performed by a unitary transformation. If the state of a qubit is represented by a column vector then a unitary transformation $U$ can be represented by a complex-value matrix such that $UU^\dagger = U^\dagger U = I$ or $U^\dagger = U^{-1}$, where $U^\dagger$ is the conjugate transpose of $U$. $U$ acts on the Hilbert space $H$ transforming a state $|\psi\rangle$ to a state $|\psi^\prime\rangle$ by a matrix multiplication such that $|\psi^\prime\rangle = U|\psi\rangle$. There are some common quantum gates: the Hadamard gate $H$, the identity gate $I$, the Pauli gates $X$, $Y$, and $Z$, and the controlled-NOT gate $CX$. Note that the $CX$ gate performs on two qubits, while the remaining gates perform on a single qubit.

Their matrix representations are as follows:

$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$, $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

For example, the Hadamard gate on a single qubit performs the mapping $|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The controlled-NOT gate on pairs of qubits performs the mapping $|00\rangle \mapsto |00\rangle$, $|01\rangle \mapsto |01\rangle$, $|10\rangle \mapsto |10\rangle$, $|11\rangle \mapsto |10\rangle$, which can be understood as inverting the second qubit (referred to as the target) if and only if the first qubit (referred to as the control) is one. For the sake of simplicity, we do not take the Pauli gate $Y$ into account in this present paper because it is not used in our case study.

A quantum measurement is described as a collection $\{M_m\}$ of measurement operators, where the indices $m$ refer to the measurement outcomes. It is required that the measurement operators satisfy $\sum_m M_m^\dagger M_m = I$. If the state of a quantum system is $|\psi\rangle$ before the measurement, then the probability for the result $m$ is as follows:

$p(m) = \langle \psi | M_m^\dagger M_m | \psi \rangle$.

The state of the quantum system after the measurement is $\sqrt{p(m)}|\psi\rangle$ provided that $p(m) > 0$. For example, if a qubit is in state $\alpha|0\rangle + \beta|1\rangle$ and measuring with $\{M_0, M_1\}$ operators, we have the result 0 with probability $|\alpha|^2$ at the post-measurement state $|0\rangle$ and the result 1 with probability $|\beta|^2$ at the post-measurement state $|1\rangle$, where $M_0 = |0\rangle \langle 0|$ and $M_1 = |1\rangle \langle 1|$.

For multiple qubits, we use the tensor product of Hilbert spaces. Let $H_1$ and $H_2$ be two Hilbert spaces. Their tensor product $H_1 \otimes H_2$ is defined as a vector space consisting of linear combinations of the vectors $|\psi_1\psi_2\rangle = |\psi_1\rangle |\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, where $|\psi_1\rangle \in H_1$ and $|\psi_2\rangle \in H_2$. Systems of two or more qubits may be in entangled states, meaning that states of qubits are correlated and inseparable. For example, we consider a measurement of the first qubit of the entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The result is 0 with probability $\frac{1}{2}$ leaving its state $|00\rangle$ or 1 with probability $\frac{1}{2}$ leaving its state $|11\rangle$. In either case, a subsequent measurement of the second qubit gives a non-probabilistic result, which is immediate to the result of the first measurement before. Entanglement shows that an entangled state of two qubits cannot be expressed as a tensor product of single-qubit states. We can use $H$ and $CX$ gates to create entangled states as follows: $CX((H \otimes I)|00\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

B. Kripke Structures

A Kripke structure $K$ is $\langle S, I, T, A, L \rangle$, where $S$ is a set of states, $I \subseteq S$ is the set of initial states, $T \subseteq S \times S$ is a left-total binary relation over $S$, $A$ is a set of atomic propositions and $L$ is a labeling function whose type is $S \rightarrow 2^A$. Each element $(s, s') \in T$ is called a state transition from $s$ to $s'$ and $T$ may be called the state transitions (with respect to $K$). For a state $s \in S$, $L(s)$ is the set of atomic propositions that hold in $s$. A path $\pi$ is an infinite sequence $s_0, \ldots, s_i, s_{i+1}, \ldots$ such that $s_i \in S$ and $(s_i, s_{i+1}) \in T$ for each $i$. We use the following notation for paths: $\pi^i \triangleq s_i, s_{i+1}, \ldots$, where $\triangleq$ is used as “be defined as.” $\pi^i$ is obtained by deleting the first $i$ states $s_0, s_1, \ldots, s_{i-1}$ from $\pi$. Let $P$ be the set of all paths. $\pi$ is called a computation if $\pi(0) \in I$. Let $C$ be the set of all computations.

The syntax of a formula $\varphi$ in LTL for $K$ is as follows:

$\varphi ::= T \mid p \mid \neg \varphi \mid \varphi \land \varphi \mid \bigcirc \varphi \mid \varphi U \varphi$

where $p \in A$. Let $F$ be the set of all formulas in LTL for $K$. An arbitrary path $\pi \in P$ of $K$ and an arbitrary LTL formula $\varphi \in F$ of $K$, $K, \pi \models \varphi$ is inductively defined as follows:

- $K, \pi \models T$
- $K, \pi \models p$
• $K, \pi \models p$ iff $p \in L(\pi(0))$
• $K, \pi \models \neg \varphi_1$ iff $K, \pi \not\models \varphi_1$
• $K, \pi \models \varphi_1 \land \varphi_2$ iff $K, \pi \models \varphi_1$ and $K, \pi \models \varphi_2$
• $K, \pi \models \bigcirc \varphi_1$ iff $K, \pi^1 \models \varphi_1$
• $K, \pi \models \varphi_1 \cup \varphi_2$ iff there exists a natural number $i$ such that $K, \pi^i \models \varphi_2$ and for all natural numbers $j < i$, $K, \pi^j \models \varphi_1$

where $\varphi_1$ and $\varphi_2$ are LTL formulas. Then, $K \models \varphi$ iff $K, \pi \models \varphi$ for each computability $\pi \in C$ of $K$. $\bigcirc$ and $\cup$ are called the next temporal connective and the until temporal connective, respectively.

In this paper, a state is expressed as a braced associative-commutative (AC) collection of name-value pairs. The order of elements is not relevant in AC collections, such as sets. AC collections are called soups, and name-value pairs are called observable components. That is, a state is expressed as a braced soup of observable components. The juxtaposition operator is used as the constructor of soups. Let $oc_1, oc_2, oc_3$ be observable components, and then $oc_1 \circ oc_2 \circ oc_3$ is the soup of those three observable components. Since the order is irrelevant because of AC, $oc_1 \circ oc_2 \circ oc_3$ is the same as some others, such as $oc_3 \circ oc_2 \circ oc_1$. A state is expressed as $\{oc_1 \circ oc_2 \circ oc_3\}$. In this paper, rewrite rules are used to specify state transitions. Concretely, we use Maude [6], a programming/specification language based on rewriting logic. Maude makes it possible to specify complex systems flexibly and is also equipped with an LTL model checker.

III. Symbolic Reasoning

This section introduces some terms used in our symbolic reasoning and a set of laws used to reduce terms.

A. Terms

Terms are built from scalars and basic vectors with some constructors.

- Scalars are complex numbers. We extend rational numbers supported in Maude to deal with complex numbers. Some constructors for scalars, such as multiplication, fraction, addition, conjugation, absolute, power, and square root are formalized, but we do not mention them here to make the paper concise.
- Basic vectors are the standard basis written in Dirac notation as $|0\rangle$ and $|1\rangle$.
- Constructors for matrices consist of scalar multiplication of matrices $\cdot$, matrix product $\times$, matrix addition $+$, tensor product $\otimes$, and the conjugate transpose $A^\dagger$ of a matrix $A$.

In Dirac notation, $\langle 0 |$ is the dual of $|0\rangle$ such that $\langle 0 | = |0\rangle$ and $|0\rangle \langle 0 | = \langle 0 |$. Similarly for $|1\rangle$. The terms $|j\rangle \times |k\rangle$ and $|j\rangle \otimes |k\rangle$ may be written shortly as $|j\rangle |k\rangle$ and $|j\rangle \langle k|$ for any $j, k \in \{0, 1\}$. By using these notations, we can intuitively explain how quantum operations work. For example, the $X$ gate performs mapping $|0\rangle \rightarrow |1\rangle$ and $|1\rangle \rightarrow |0\rangle$. Therefore, we formalize the $X$ gate as $|0\rangle \langle 1 | + |1\rangle \langle 0 |$ in Maude instead of using explicitly the matrix representation

$$
\begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}
$$

It is immediate that $X |0\rangle = |1\rangle \langle 0 | + |0\rangle \langle 1 | = |1\rangle$ because of the use of some laws in Table I and similarly for $X |1\rangle$.

We conventionally formalize some basic matrices $B_i$ for $i \in [0..3]$ as follows:

$$
B_0 = |0\rangle \langle 0 |, \quad B_1 = |0\rangle \langle 1 |, \\
B_2 = |1\rangle \langle 0 |, \quad B_3 = |1\rangle \langle 1 |
$$

The $CX$, $X$, $Z$, and $H$ gates are then a linear combination of the matrices $B_i$ as follows:

$$
CX = B_0 \otimes I_2 + B_3 \otimes X, \\
X = B_1 + B_2, \quad Z = B_1 + (-1) \cdot B_3, \\
H = \frac{1}{\sqrt{2}} \cdot B_0 + \frac{1}{\sqrt{2}} \cdot B_1 + \frac{1}{\sqrt{2}} \cdot B_2 + (-\frac{1}{\sqrt{2}}) \cdot B_3.
$$

B. Laws

We use a set of laws in Table I derived from the properties of quantum mechanics and basic matrix operations and thus they are immediately sound (see their proofs in Coq in [1]). Because $|0\rangle$ and $|1\rangle$ can be viewed as $2 \times 1$ matrices, then the laws actually describe matrix calculations with Dirac notation, zero and identity matrices, and scalars. These laws are described by equations in Maude and are used to automatically reduce terms until no more matrix operation is applicable. Some laws dedicated to simplifying the expressions about complex numbers are also formalized in Maude by means of equations, but we do not describe them here to make the paper concise.

For example, we would like to reduce the term

$$
H \times |0\rangle
$$

To check whether its result is $\frac{1}{\sqrt{2}} \cdot |0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes |1\rangle$. The term says that the $H$ gate acts on the first qubit followed by the $CX$ gate where the control and target bits are the first and second qubits, respectively. The simplification of the term goes as follows:

$$
H \times |0\rangle
= (\frac{1}{\sqrt{2}} \cdot B_0 + \frac{1}{\sqrt{2}} \cdot B_1 + \frac{1}{\sqrt{2}} \cdot B_2 + (-\frac{1}{\sqrt{2}}) \cdot B_3) \times |0\rangle
= \frac{1}{\sqrt{2}} \cdot B_0 \times |0\rangle + \frac{1}{\sqrt{2}} \cdot B_1 \times |0\rangle + \frac{1}{\sqrt{2}} \cdot B_2 \times |0\rangle + (-\frac{1}{\sqrt{2}}) \cdot B_3 \times |0\rangle
= \frac{1}{\sqrt{2}} \cdot |0\rangle \times |0\rangle + \frac{1}{\sqrt{2}} \cdot |0\rangle \times |1\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \times |0\rangle + (-\frac{1}{\sqrt{2}}) \cdot |1\rangle \times |1\rangle
= \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle
$$

This reduces the term to

$$
\langle H \otimes I \rangle \times (|0\rangle \otimes |0\rangle)
$$

$$
= \langle H \times |0\rangle \otimes |0\rangle \rangle
= \langle \langle \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes |0\rangle \rangle
= \frac{1}{\sqrt{2}} \cdot |0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes |0\rangle
$$

$$
CX \times ((H \otimes I) \times (|0\rangle \otimes |0\rangle))
$$

$$
= (B_0 \otimes I + B_3 \otimes X) \times (\frac{1}{\sqrt{2}} \cdot |0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes |1\rangle)
$$

Using the laws, the term is reduced to a normal form that is a linear combination of the tensor product of the standard
basis with scalars. The whole process is conducted automatically in Maude and the result is the same as expected. The key idea is to reduce the matrix multiplication in the form of \( |i\rangle \langle j| \) into a scalar and simplify the matrix representation by absorbing ones and eliminating zeros (see the law with label L3). In this manner, our symbolic reasoning about matrices can be conducted automatically by rewriting in Maude instead of explicitly calculating matrix multiplications.

IV. QUANTUM TELEPORTATION PROTOCOL

We use quantum teleportation protocol [7] as a case study to demonstrate how our symbolic reasoning can be used to model check quantum circuits in Maude. The protocol takes advantage of entanglement in quantum mechanics to send an unknown quantum state \( |\psi\rangle \) from a sender to a receiver by using only three qubits and two classical bits. The circuit depicted in Fig. 1 shows how the protocol works. The single wires denote qubits referred to as \( q_0 \), while the double wires denote classical bits referred to as \( c_i \). The sender acts on \( q_0 \) and \( q_1 \), and the receiver acts on \( q_2 \) as follows:

- First, we prepare an unknown state \( |\psi\rangle = \alpha |0\rangle + \beta |1\rangle \) at \( q_0 \), where \( \alpha \) and \( \beta \) are complex numbers such that \( |\alpha|^2 + |\beta|^2 = 1 \). Initially, \( q_1 \) and \( q_2 \) are in the state \( |0\rangle \).
- Second, we apply a sequence of quantum gates to manipulate three qubits. We first apply the \( H \) gate on \( q_1 \) followed by the \( CX \) gate on \( q_1 \) and \( q_2 \) in order to make an entangled state shared between the sender and the receiver. The sender then applies the \( CX \) gate on \( q_0 \) and \( q_1 \) followed by the \( H \) gate on \( q_0 \).
- Third, we measure the qubits \( q_0 \) and \( q_1 \) and immediately obtain two classical outcomes \( 0 \) or \( 1 \) stored in \( c_0 \) and \( c_1 \), respectively.
- Fourth, we conditionally apply single-qubit \( Z \) and \( X \) gates on \( q_2 \) depending on the two classical bits in \( c_0 \) and \( c_1 \). Concretely, we use the \( X \) gate if \( c_1 \) equals \( 1 \) and follow by the \( Z \) gate if \( c_0 \) equals \( 1 \).

At the end, the receiver will have \( |\psi\rangle \) and the sender will not have anymore. We would verify whether the sender can send correctly an arbitrary unknown quantum state to the receiver at the end by using our symbolic model checking.

V. FORMAL SPECIFICATION

A. Formalization of Qubits, Gates, and Measurements

Qubits are formalized as the linear combination of tensor product of the standard basis in Dirac notation with scalars and similarly for quantum gates. Because \( |0\rangle \) and \( |1\rangle \) can be viewed as \( 2 \times 1 \) matrices, then qubits and quantum gates are basically matrices. Quantum gates act on qubits (a quantum state) formalized as a matrix multiplication with a deterministic transition in Maude. In this paper, we only consider projective measurements on the standard basis, and thus the measurement operators are \( \{ M_0, M_1 \} \). A measurement of a single qubit in a quantum state is formalized by two state transitions with probabilities \( p(m) \) for \( m \in \{0,1\} \), making a non-deterministic probabilistic transition. Each of the two transitions shows how its measurement operator acts on the single qubit in a state and is formalized similarly as quantum gates, however, with respect to the probabilities.

B. A Generic Formalization of Quantum Circuits

A whole quantum state is formalized as a collection of qubits associated with indices in circuits, where each element is one of the forms as follows:

- \( \langle c | i \rangle = |\psi\rangle \) denote a single qubit in state \( |\psi\rangle \) at \( q_i \),
- \( \langle c | i, \ldots, j \rangle = |\psi\rangle \) denote an entangled state in state \( |\psi\rangle \) at \( q_i, \ldots, q_j \) where the order of \( i, \ldots, j \) is relevant.

Classical bits are formalized as a map from indices in circuits to Boolean values, where each entry is in the form of \( i \mapsto b \), meaning that the value of the classical bit stored at \( c_i \) is \( b \) whose value is either \( 0 \) or \( 1 \).

A sequence of quantum gates, measurements, and conditional gates in a quantum circuit is formalized as a list of actions in which each action is one of the forms as follows:
\[ c[1] == b \ ? \ AL \text{ checks if the classical bit at } c_i \text{ equals } b, \text{ then a list } AL \text{ of actions is executed.} \]

Based on the actions formalized above, we can describe the circuit for quantum teleportation protocol as follows:

\[
\begin{align*}
H(1) & \ CX(1, 2) \ CX(0, 1) \ H(0) \ M(0) \ M(1) \\
(c[1] == 1 \ ? \ X(2)) & \ (c[0] == 1 \ ? \ Z(2))
\end{align*}
\]

Let \( K_C \) be the Kripke structure formalizing a quantum circuit. There are five kinds of observable components in our formalization as follows:

- \( (\text{qstate: } q) \) denotes the whole quantum state \( q \).
- \( (\text{bits: } b) \) denotes the classical bits obtained from measurements and stored in a bit map \( b \).
- \( (\text{prob: } p) \) denotes the probability \( p \) at the current quantum state.
- \( (\text{actions: } a) \) denotes the action list \( a \), guiding us on how the circuit works.
- \( (\text{isEnd: } b) \) denotes termination with Boolean flag \( b \).

Each state in \( S_C \) is expressed as \( \{ \text{obs} \} \), where \( \text{obs} \) is a soup of one quantum observable component, one probabilistic observable component, one observable component, and one isEnd observable component.

\( T_C \) consists of 10 rewrite rules in our formalization. Let \( OCs \) be a Maude variable of observable component soups, \( Q \) and \( Q' \) be Maude variables of whole quantum states, \( BM \) be a Maude variable of bit maps, \( Prob \) and \( Prob' \) be Maude variables of scalars, \( AL \) and \( AL' \) be Maude variables of action lists, \( B \) be a Maude variable of Boolean values, and \( N, N_1, N_2 \) are Maude variables of natural numbers.

The first five rewrite rules are as follows:

\[
\begin{align*}
&\text{rl [I] : } \{ (\text{qstate: } q) \ (\text{actions: } (I(N) AL)) \ \text{OCs} \} \Rightarrow \{ (\text{qstate: } q) \ (\text{actions: } AL) \ \text{OCs} \} . \\
&\text{rl [X] : } \{ (\text{qstate: } q) \ (\text{actions: } (X(N) AL)) \ \text{OCs} \} \Rightarrow \{ (\text{qstate: } q') \ (\text{actions: } AL) \ \text{OCs} \} \\
&\text{if } Q' := (Q).X(N) . \\
&\text{rl [Z] : } \{ (\text{qstate: } q) \ (\text{actions: } (Z(N) AL)) \ \text{OCs} \} \Rightarrow \{ (\text{qstate: } q') \ (\text{actions: } AL) \ \text{OCs} \} \\
&\text{if } Q' := (Q).Z(N) . \\
&\text{rl [H] : } \{ (\text{qstate: } q) \ (\text{actions: } (H(N) AL)) \ \text{OCs} \} \Rightarrow \{ (\text{qstate: } q') \ (\text{actions: } AL) \ \text{OCs} \} \\
&\text{if } Q' := (Q).H(N) . \\
&\text{rl [CX] : } \{ (\text{qstate: } q) \ (\text{actions: } (CX(N_1, N_2) AL)) \ \text{OCs} \} \Rightarrow \{ (\text{qstate: } q') \ (\text{actions: } AL) \ \text{OCs} \} \\
&\text{if } Q' := (Q).CX(N_1, N_2) .
\end{align*}
\]

The rules \( I, X, Z, H \), and \( CX \) simulate how the \( I, X, Z, H, \) and \( CX \) gates act on the whole quantum state in quantum observable component if its action appears in actions observable component, respectively.

The next two rewrite rules are as follows:

\[
\begin{align*}
&\text{rl [M0] : } \{ (\text{qstate: } q) \ (\text{actions: } (M(N) AL)) \ (\text{prob: } Prob) \ (\text{bits: } BM) \ \text{OCs} \} \\
&\Rightarrow \{ (\text{qstate: } q') \ (\text{actions: } AL) \ (\text{prob: } (\text{Prob} .\text{if } c[N] == N_2 \ ? \ AL')) \ (\text{bits: } (\text{insert}(N, 1, BM) \ \text{OCs} \} . \\
&\text{rl [M1] : } \{ (\text{qstate: } q) \ (\text{actions: } (M(N) AL)) \ (\text{prob: } (\text{bits: } BM) \ \text{OCs} \} \\
&\Rightarrow \{ (\text{qstate: } q') \ (\text{actions: } AL) \ (\text{prob: } (\text{if } c[N] == N_2 \ ? \ AL' \ ? \ Prob')) \ (\text{bits: } (\text{insert}(N, 1, BM) \ \text{OCs} \} .
\end{align*}
\]

The rules M0 and M1 say that we measure the qubit at index \( N \) with the measurement operators \( M_0 \) and \( M_1 \), respectively; the classical outcomes are stored accordingly into the bit map in bits observable component; the probabilities and the post-measurement states are also updated in prob and qstate observable components, respectively. These two rules make a non-deterministic probabilistic transition when measuring a single qubit.

The next rewrite rule describes how to conditionally perform the next actions based on classical bits obtained from measurements if applicable.

\[
\begin{align*}
&\text{rl [cIf] : } \{ (\text{qstate: } q) \ (\text{bits: } ((N |-> N_1, BM) \ (\text{actions: } ((c[N] == N_2 \ ? \ AL') AL)) \ \text{OCs} \} \\
&\Rightarrow \{ (\text{qstate: } q) \ (\text{bits: } ((N |-> N_1, BM)) \ (\text{actions: } ((\text{if } (N_1 == N_2 \ ? \ AL' \ ? \ Prob')) \ (\text{bits: } (\text{insert}(N, 1, BM) \ \text{OCs} \} .
\end{align*}
\]

This rule says that if \( c[N] == N_2 \ ? \ AL' \ ? \ Prob' \) is in the action list and the classical bit \( N_1 \) at index \( N \) equals the conditional value \( N_2 \), then the action list \( AL' \ ? \ Prob' \) is prepended to the action list \( AL \) in actions observable component to be executed next; otherwise, it is ignored.

The last two rules are as follows:

\[
\begin{align*}
&\text{rl [end] : } \{ (\text{actions: } nil) \ (\text{isEnd: } false) \ \text{OCs} \} \Rightarrow \{ (\text{actions: } nil) \ (\text{isEnd: } true) \ \text{OCs} \} . \\
&\text{rl [stutter] : } \{ (\text{isEnd: } true) \ \text{OCs} \} \Rightarrow \{ (\text{isEnd: } true) \ \text{OCs} \}.
\end{align*}
\]

The rule end marks the termination if the action list is nil, meaning no more action. Meanwhile, the rule stutter is necessary to make \( T_C \) total when isEnd observable component is true.

### C. Formalization of Quantum Teleportation Protocol

To formalize quantum teleportation protocol, let \( I_C \) consist of only one initial state as follows:

\[
\begin{align*}
&\{ (\text{isEnd: false}) \ (\text{prob: } 1) \ (\text{bits: } empty) \} \\
&\{ (\text{qstate: } q[0]: a, \ 0? + b, \ 1?) \} \\
&\{ (\text{qstate: } q[1]: 0?) \ (\text{qstate: } q[2]: 0?) \} \\
&\{ (\text{actions: } H(1) \ CX(1,2) \ CX(0,1) \ H(0) \ M(0) \ M(1)) \ (c[1] == 1 \ ? \ X(2)) \ (c[0] == 1 \ ? \ Z(2)) \}
\end{align*}
\]

where \( a \) and \( b \) are Maude constants of scalars denoting arbitrary scalars such that \( |a|^2 + |b|^2 = 1 \). Initially, isEnd observable component is false, prob observable component is one, qstate is a symbolic state as the input state of the protocol, actions observable component contains the action list describing how the protocol works. For other
protocols, we only need to formalize the initial quantum state and the action list in the initial state of \( I_C \), while we can definitely reuse \( S_C \) and \( T_C \) in \( K_C \), making our formalization as a general framework to formally specify quantum circuits.

VI. SYMBOLIC MODEL CHECKING

Let TELEPORT be the specification of the quantum teleportation protocol, \( \text{init} \) be the initial state for TELEPORT, \( \text{qstate} \) and \( \text{qubitAt} \) be functions to get the whole quantum state from the initial state and to get a single qubit at some index, respectively. To model check that \( K_C \) satisfies some desired properties, we specify \( A_C \) and \( L_C \). \( A_C \) has one atomic proposition \( \text{isSuccess} \). \( L_C \) is specified as follows:

\[
\text{eq} \{ (\text{isEnd}: \text{true}) \ (\text{qstate}: Q) \ (\text{prob}: \text{Prob}) \}
\]

\[
\text{OCs} \models \text{isSuccess}
\]

\[
= \text{Prob} > 0 \ \text{implies}
\]

\[
\text{qubitAt}(Q, 2) == \text{qubitAt}(\text{qstate}(\text{init}), 0).
\]

\[
\text{eq} \{ \text{OCs} \} \models \text{PROP} = \text{false \ owise}.
\]

The two equations say that \( \text{isSuccess} \) holds at a state if the state contains \( (\text{isEnd}: \text{true}), (\text{qstate}: Q) \), and \( (\text{prob}: \text{Prob}) \) such that the condition \( \text{qubitAt}(Q, 2) == \text{qubitAt}(\text{qstate}(\text{init}), 0) \) holds whenever \( \text{Prob} > 0 \), meaning that the qubit received by the receiver at the end is equal to the qubit sent by the sender at the beginning with a non-zero probability. Let a LTL formula \( \text{teleProp} \) be defined as True \( U \) \( \text{isSuccess} \), where \( U \) is the temporal until operator.

We model check that \( K_C \) satisfies \( \text{teleProp} \) from the initial state \( \text{init} \) in Maude as follows:

\[
\text{red} \ \text{modelCheck}(\text{init}, \text{teleProp}).
\]

No counterexample is found in just a few moments; thus, \( K_C \) satisfies \( \text{teleProp} \). In other words, we successfully verify the correctness of the quantum teleportation protocol by using our symbolic model checking approach. Because our formalization considers the probabilities at each state, then we are able to check not only qualitative properties but also quantitative properties with Maude LTL model checker.

VII. RELATED WORK

There are several studies [8], [9] in the early work of formal specification and verification of quantum protocols. For example, Gay et al. provide a way to use classical model checkers (e.g., PRISM - a probabilistic model checker) to analyze quantum protocols. They give each quantum state a unique number and the transition from a unique number to another unique number models the action of quantum gates and measurements. Their approach needs to enumerate states and calculate the state transitions in advance and then encode them into a PRISM specification. Although they develop a so-called PRISMGEM tool to automate this, their approach is impractical in reality and only supports two or three qubits because of the exponential growth of the number of states. Our approach does not need to enumerate such states in advance because a quantum state is directly formalized in Dirac notation with scalars. Moreover, rewrite rules are used to formalize the action of quantum gates and measurements, making our approach feasible to deal with more qubits.

Our symbolic approach to model checking quantum circuits is inspired by the work [1]. However, their approach is oriented to theorem proving, not model checking. They also use Dirac notation with a small set of laws to specify quantum states, quantum gates, measurements, and reasoning about quantum circuits in Coq, an interactive theorem prover. However, they usually require human users to provide necessary lemmas to complete their proofs, which are not easy tasks in general. Meanwhile, our approach is fully automatic and does not need any intervention from human users. Moreover, our implementation can be used as a general framework to formally specify and verify quantum circuits in a symbolic way in Maude.

VIII. CONCLUSION

We have proposed a symbolic approach to model checking quantum circuits by using a set of laws from quantum mechanics and basic matrix operations with Dirac notation. We have analyzed the quantum teleportation protocol as a case study to demonstrate the usefulness of our approach. Moreover, our implementation developed in Maude can be used as a general framework to formally specify and verify quantum circuits using our symbolic model checking approach. Our formalization takes the probabilities into account, and then we can tackle both qualitative and quantitative properties with the built-in LTL model checker in Maude. As one piece of our future work, we would like to conduct more case studies to demonstrate the usefulness of our approach.

REFERENCES


Session SEKE3: Software Engineering and Knowledge Engineering
Using Z3 for Formal Modeling and Verification of FNN Global Robustness

Yihao Zhang*, Zeming Wei*, Xiuyue Zhang*, †, Meng Sun‡
*School of Mathematical Sciences, Peking University, Beijing, China
†Corresponding author.
‡Oxford, UK.

Abstract—While Feedforward Neural Networks (FNNs) have achieved remarkable success in various tasks, they are vulnerable to adversarial examples. Several techniques have been developed to verify the adversarial robustness of FNNs, but most of them focus on robustness verification against the local perturbation neighborhood of a single data point. There is still a large research gap in global robustness analysis. The global-robustness verifiable framework DeepGlobal has been proposed to identify all possible Adversarial Dangerous Regions (ADRs) of FNNs, not limited to data samples in a test set. In this paper, we propose a complete specification and implementation of DeepGlobal utilizing the SMT solver Z3 for more explicit definition, and propose several improvements to DeepGlobal for more efficient verification. To evaluate the effectiveness of our implementation and improvements, we conduct extensive experiments on a set of benchmark datasets. Visualization of our experiment results shows the validity and effectiveness of the approach.

Index Terms—Feedforward Neural Networks, Global Robustness Verification, Social Aspects of Artificial Intelligence

I. INTRODUCTION

Feedforward Neural Networks (FNNs) have achieved remarkable success in various fields. Despite their success, the existence of adversarial examples [3] highlights the vulnerability of FNNs and raises concerns about their safety in critical domains. Adversaries can easily deceive FNNs by introducing small and imperceptible perturbations to natural inputs, resulting in erroneous predictions. Although adversarial training [5] is considered the most effective approach for training models that are resistant to adversarial attacks, it still has a serious weakness, i.e., the lack of formal guarantees of the robustness.

To solve this problem, an avenue of research involves formally modeling and verifying the robustness of given models [4]. These methods can provide provable verification of local robustness, which pertains to specific input samples. However, simply evaluating a model’s local robustness against a test set cannot provide global robustness. To explore global robustness verification, [7] proposed to approximate the global-robust radius utilizing the Hamming distance. Despite this, the approach in [7] still depends on a test set, which is not entirely satisfactory for global robustness verification.

Another inherent challenge in neural network verification is the computational complexity. The number of activation patterns, that is the potential activation status of non-linear neurons, can be of an exponential order of magnitude. Therefore, it is not practical to cover all possible patterns as the model size increases rapidly nowadays. To address this issue, existing approaches used linear relaxation [10] and abstract interpretation [2] techniques for adversarial training and verification. However, these methods all focus on local robustness. To achieve global robustness analysis, DeepGlobal [8] was proposed to facilitate global robustness verification of FNNs. It introduces a novel neural network architecture, Sliding Door Network (SDN), where all adversarial regions can be more efficiently generated. As rigorous formalization is crucial for safety verification, further steps must be taken to formally prove the global robustness of the new neural network SDN.

In this paper, we build upon the DeepGlobal framework and use the SMT solver Z3 [6] to provide a complete specification and implementation of the framework. Specifically, we provide formal definitions of DeepGlobal and algorithms with several improvements for more efficient generation of adversarial dangerous regions. To demonstrate how the Z3 solver can be applied to verify the global robustness of FNNs, we conduct extensive experiments on the MNIST and Fashion-MNIST datasets. The code is available at https://github.com/weizeming/Z3_for_Verification_of_FNN_Global_Robustness.

The paper is organized as follows. In Section II, we provide preliminaries on Feedforward Neural Networks (FNNs), Adversarial Dangerous Regions (ADRs) and Sliding Door Activation (SDA). We introduce the Z3 specification for FNNs and SDNs in Section III. In Section IV, we further show the Z3 specifications of SDNs and ADRs, which provides an explicit definition of the DeepGlobal framework. Furthermore, we present algorithmic implementation details in Section V. Section VI concludes the paper.

II. PRELIMINARIES

We consider a $K$-classification neural network $F : \mathcal{X} \to \mathcal{Y}$, which maps an input space $\mathcal{X} \subset \mathbb{R}^d$ to an output space $\mathcal{Y} = \{1, 2, \cdots, K\}$. Let $\hat{F}(x)$ denote the ground-truth classification result for $x \in \mathcal{X}$ as determined by a human expert.

We define a FNN $f$ as a tuple $(m, N, W, A)$, where $m$ is the number of layers in $f$, $N = (n_1, n_2, \cdots, n_m)$ is a vector specifying the number of neurons in each layer, $W = (w_1, b_1, \cdots, w_m, b_m)$ is a set of parameters for $f$, where $w_i \in \mathbb{R}^{n_{i-1} \times n_i}$ and $b_i \in \mathbb{R}^{n_i}$, and $A = (a_1, a_2, \cdots, a_m)$ is a
set of activation functions for each layer, where \( a_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i} \). Thus we have \( f(x) = a_m(w_m \cdot \cdots (w_1 \cdot x + b_1) + b_m) \).

Note that the input dimension of the FNN \( f \) satisfying that \( n_0 = d \), and the output dimension \( n_m = K \). Given \( f(x) = (f(x)_1, f(x)_2, \cdots, f(x)_K) \), the FNN returns its prediction \( F(x) = \arg \max_i f(x)_i \).

Adversarial examples [9] are inputs with a small perturbation \( \delta \) added to a benign sample \( x \) such that the model misclassifies the perturbed sample \( F(x + \delta) \neq F(x) \). The perturbation \( \delta \) is constrained by an \( \ell_p \)-norm ball, such as \( |\delta|_p \leq \epsilon \). The concept of Adversarial Dangerous Regions (ADRs) is introduced to characterize global robustness. ADRs present the property that each neuron is near the decision boundary and the samples in it have clear semantics. We can formally model these conditions as

\[
\begin{align*}
\Delta^*_{F} := \{ x | \exists y, i \neq j : ||x - y||_p \leq \epsilon, \\
F(y)_i = & F(y)_j \geq F(y)_k, (\forall k \neq i, j), \tilde{F}(y) = i \}.
\end{align*}
\]

In the DeepGlobal framework, the SDA function is proposed to reduce the number of activation patterns in the SDNs. SDA divides the neurons in each layer \((h_{i,1}, h_{i,2}, \cdots, h_{i,n_i})\) into groups of \( k \) neurons. Let the divided groups be denoted as \( G_{h,0}, G_{h,1}, \cdots, G_{h,l} \), where \( l = \frac{n_i}{k} \). SDA finds the first group \( G_{h,\text{Act}} \), in which pre-activated neurons are all positive, from left to right. This group presents the property that each neuron within it is active and is preferred for activation. Therefore, SDA names this group the Active Door and multiplies it by a constant \( \alpha > 1 \) to stimulate the active neurons as activation. Additionally, SDA searches for an Inactive Door \( G_{h,\text{Ina}} \), in which neurons are all negative and multiplies them by 0 to penalize the inactive neurons. The remaining \( l - 2 \) doors are named Trivial Doors, which SDA neither activates nor deactivates but retains their values after activation. SDN leverages SDA to achieve comparable accuracy to general FNNs while significantly reducing the magnitude of activation patterns, making it an efficient candidate for verification.

III. FORMALIZATION OF SLIDING DOOR NETWORKS

A. Formulation of FNNs

As the concept of SDNs is based on FNNs, we first demonstrate how to use Z3 [6] to formally model a given FNN. We assume the FNN configuration (e.g., input dimension \( d \)) has already been declared and represent each variable in the input, hidden, and output layers as a ‘Real’ object in Z3:

\[
\begin{align*}
\text{Input} &= \{ \text{Real}(f^*_{x\rightarrow (\cdot)\#}) \text{ for } i \text{ in range(d)} \} \\
\text{Hidden} &= \{ \text{Real}(f^*_{h_{\cdot\rightarrow (\cdot)\#}}) \text{ for } j \text{ in range(N(i))} \text{ for } i \text{ in range(m-1)} \} \\
\text{Output} &= \{ \text{Real}(f^*_{y\rightarrow (\cdot)\#}) \text{ for } i \text{ in range(K)} \}
\end{align*}
\]

In this way, the input and output variables are named ‘\( x^* \)’, ‘\( y^* \)’ respectively, where \( i \) indicates the \( i \)-th input (output) variable (counting from zero). For \( 0 \leq i \leq m - 2 \), the \( j \)-th hidden variable in the \( i \)-th layer is named ‘\( h_{i,j}^* \)’ (counting from zero), and note that the \( m - 1 \)-th layer is the output layer.

The constraints between input, hidden, and output layers highly depend on the activation patterns. Therefore, we can only model the constraints for each potential activation pattern respectively, which include four parts:

**The constraints on input domain \( X \).** Taking the MNIST dataset as example, since each pixel value is restricted to \([0,1]\), let \( s \) be the initialized solver to be used later, we have

\[
\begin{align*}
s &= \text{Solver}() \\
s.add([\text{Input}[i] >= 0 \text{ for } i \text{ in range(d)])} \\
s.add([\text{Input}[i] <= 1 \text{ for } i \text{ in range(d)])}
\end{align*}
\]

We denote these constraints as \( C_{\text{input}} \).

**The relation between adjacent layers under given activation patterns.** The forward-pass from \( h_{i-1} \) (the \( i-1 \)-th layer) to \( y \) (the \( i \)-th layer) can be formulated as \( h_i = a_i(w_i \cdot h_{i-1} + b_i) \). For the sake of simplicity, we introduce variables ‘\( h_{i,j}^* \)’ for the pre-activate neurons \( h_i = w_i \cdot h_{i-1} + b_i \).

\[
\begin{align*}
\text{_Hidden} &= \{ \text{Real}(f^*_{h_{\cdot\rightarrow (\cdot)\#}}) \text{ for } j \text{ in range(N(i))} \text{ for } i \text{ in range(m-1)} \}
\end{align*}
\]

In this way, we can simplify the constraint from layer \( h_i \) to \( y \) with the aid of \( h_i^* \):

\[
\begin{align*}
s.add([\text{Hidden}[i][j] == \text{Sum}([W[i][j][k] * \text{Hidden}[i-1][k]) \text{ for } k \text{ in range(n(i-1))}] + [B[i][j])} \\
s.add([\text{Hidden}[i][j] == [W[i][j][j]) \text{ for } i \text{ in range(m-1))}) \text{ for } k \text{ in range(k)}]
\end{align*}
\]

**The activation condition of the given activation patterns.** We defer this part in Section IV after introducing the SDA functions.

**The objective property.** For example, if we want to identify samples from class \( i \), which are also near the decision boundary with class \( j \), the constraints should be formulated as \( f(x)_i = f(x)_j, f(x)_k > f(x)_k \). This can be expressed with Z3 constraints as:

\[
\begin{align*}
s.add([\text{Output}[i] == \text{Output}[j])} \\
& \text{for } k \text{ in range(k)}: \\
& \text{if } k == i \text{ or } k == j: \\
& \text{continue} \\
s.add([\text{Output}[i] == \text{Output}[k])
\end{align*}
\]

The specification details of adversarial dangerous regions (ADRs) is presented in Section IV. B. Formulation of SDNs

A SDN is a feedforward neural network \( f \) with the tuple \((m,N,W,A,k)\), where \( m \), \( N \), and \( W \) are inherited from the definition of FNN, and \( A = (a_1, \cdots, a_m) \) is the SDA function. The parameter \( k \) represents the number of neurons in each group. Let \( h_i \) denote the hidden variables in the \( i \)-th layer, with \( h_0 = x \) being the input and \( h_{m+1} = f(x) \) being the output. We can recursively define the mapping \( f \) as follows:

\[
\begin{align*}
\hat{h}_i &= w_i \cdot \hat{h}_{i-1} + b_i, \\
\begin{cases}
\text{Act}_i = \min_g \forall (g - 1) \cdot k < j \leq g \cdot k, \quad \hat{h}_{i,j} > 0, \\
\text{Ina}_i = \min_g \forall (g - 1) \cdot k < j \leq g \cdot k, \quad \hat{h}_{i,j} < 0.
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\hat{h}_{i,j} &= \begin{cases}
\alpha \cdot \hat{h}_{i,j} + (\text{Act}_i - 1) \cdot k < j \leq \text{Act}_i \cdot k, \\
0, \quad (\text{Ina}_i - 1) \cdot k < j \leq \text{Ina}_i \cdot k \\
\hat{h}_{i,j}, \quad \text{else},
\end{cases}
\end{align*}
\]
Note that the \( \text{Act}_j \) or \( \text{Ina}_i \) in (2) may not exist in some layers. In this case, SDN simply abandons the active or inactive door when mapping through these layers.

### IV. Complete Modeling

#### A. Modeling the Activation Conditions

As discussed in Section III-A, the specification of FNNs depends on the Activation Patterns (AP), i.e., the different configurations of active and inactive neurons in the network. For a SDN with \( m \) layers, we define an activation pattern \( \mathcal{A} = (\text{Act}_1, \text{Ina}_1, \ldots, \text{Act}_m, \text{Ina}_m) \), where \( \text{Act}_i \) and \( \text{Ina}_i \) correspond to the indices of the active and inactive doors in layer \( i \), respectively (counting from 0 to be consistent with the code). If the active or inactive door does not exist, we fill \( \text{Act}_i \) or \( \text{Ina}_i \) with \( \frac{n_i}{2} \) (the number of groups in this layer).

Therefore, given an activation pattern \( \mathcal{A} \), we give the specification of activation conditions as:

```plaintext
for i in range(m):
    if Act[i] != n[i]//k:
        s.add([Hidden[i][j] > 0 for j in range(Act[i] * k, (Act[i+1] * k))])
    else:
        if Ina[i] != n[i]//k:
            s.add([Hidden[i][j] < 0 for j in range(Ina[i] * k, (Ina[i+1] * k))])
```

The constraint is denoted as \( C_{AP}(\mathcal{A}) \) and skipped when \( \text{Act}_i \) or \( \text{Ina}_i \) is equal to \( \frac{n_i}{2} \). Note that we do not explicitly model the minimality of \( \text{Act}_i \) or \( \text{Ina}_i \), which may result in covered and common boundaries of activation regions.

The above issues are addressed by successively eliminating already-covered or common boundaries in [8]. For instance, to remove covered or common boundaries with a previous region \( \bigwedge P_j \), they conjunct each \( \neg P_j \) with \( \bigwedge R_i \) to create a new region. Using this approach, we only need to consider \( -C_{AP}(\mathcal{A}') \lor C_{AP}(\mathcal{A}) \) to remove covered and common boundaries with \( \mathcal{A}' \) for \( \mathcal{A} \).

#### B. Modeling Sliding Door Activation

Recall from Section III-A that we have modeled the linear transformation from \( h_{i-1} \) to \( h_i \). Now, we provide the formal specification of the activation function \( h_i = a_i(h_{i-1}) \), which is dependent on a given activation pattern \( \mathcal{A} \).

```plaintext
for i in range(m):
    for j in range(n[i]//k):
        if Act[i] == j: # Active Door
            s.add([Hidden[i][j+l] == alpha * Hidden[i][j] for l in range(k)])
        elif Ina[i] == j: # Inactive Door
            s.add([Hidden[i][j+l] == 0 for l in range(k)])
        else: # Trivial Door
            s.add([Hidden[i][j+l] == Hidden[i][j] for l in range(k)])
```

We denote this set of constraints (including the constraints on linear mappings) as \( C_{Forward}(\mathcal{A}) \).

#### C. Modeling the Adversarial Dangerous Regions

Recall our refined definition of ADRs in Section II, where we aim to find feasible \( y \) that satisfies the boundary condition (i.e., \( \exists i \neq j \) such that \( \forall k \neq i,j, F(y)_k = F(y)_j \geq F(y)_h \) and the meaningful condition (i.e., \( \tilde{F}(y) = i \)). In Section III-A, we present the Z3 specification for the boundary condition, which we denote as \( C_{Boundary}(i, j) \). [8] attempt to find feasible and meaningful solutions in the ADRs instead of considering the meaningful condition. Specifically, a trained autoencoder [1] is used to optimize a feasible solution \( x' \) in a given ADR, while ensuring that it remains within the same ADR. However, this optimization-based method has some limitations. For instance, the meaningful solution may not always exist for all ADRs, which is a possible scenario when all samples in the region are deemed "rubbish". Additionally, optimizing the solution along certain directions within the region can be extremely time-consuming.

Therefore, we propose a new approach that allows for more straightforward identification of meaningful samples. Note that the meaningful condition is \( \tilde{F}(y) = i \). When judging each sample by \( \tilde{F} \) (i.e., human-perception) is not practical, we can still use autoencoders as surrogate models.

For a given class \( i \in \{1, 2, \ldots, K\} \), we hope to find a meaningful region by the surrogate model \( AE \) where \( \tilde{F}(y) = i \). To achieve this, we train an autoencoder \( E(\cdot) \) and leverage it to define the center of class \( i \) as \( c_i = \frac{1}{|X_i|} \sum_{x \in X_i} E(x) \), where \( E(\cdot) \) is the encoder function, and \( X_i \) represents the samples in the training set with class \( i \), and define the prototype \( i \) as \( p_i = D(c_i) \). The prototype model for class \( i \) is decoded from the average code of samples in that class, making it a standard representation of that class. Our assumption is that any meaningful sample \( y \) with \( \tilde{F}(y) = i \) should not be significantly different from the prototype \( p_i \). To ensure this, we require \( y \) to a meaningful region \( |y - p_i|_p \leq r \), where \( r \) is a pre-specified radius. It’s worth noting that the definition of the meaningful region is fundamentally different from that of adversarial examples (see Section II), while the perturbation \( \delta \) is limited to a specific bound \( \epsilon \). Generally, \( r \) is much larger than \( \epsilon \), as all samples in this region are close to the prototype and potentially meaningful. The definition of adversarial examples is more restrictive than that of meaningful regions, as it only focuses on a small perturbed region.

Based on the above analysis, taking \( \infty \)-norm as example, we specify the meaningful condition as follows:

```plaintext
for i in range(d):
    s.add([Input[i] - P[i] < r, P[i] - Input[i] < r])
```

We denote this set of constraints as \( C_{Meaningful}(i) \). So far, we have completed all specifications for DeepGlobal in Z3. To find the feasible and meaningful sample \( y \) in the target class \( i \), which is on the decision boundary of the boundary class \( j \), with regard to activation pattern \( \mathcal{A} \), one only need to solve the following constraints in Z3:

\[
C_{Input} \land C_{AP}(\mathcal{A}) \lor C_{Forward}(\mathcal{A}) \land C_{Boundary}(i, j) \land C_{Meaningful}(i).
\]

#### V. Algorithmic Implementation Details

In this section, we demonstrate the implementation of using Z3 solver to specify the DeepGlobal framework and identify global adversarial regions.
Algorithm 1: Find feasible and meaningful solutions

Input: SDN Network \( f = (m, N, W, A, k) \); Target class \( i \); Boundary class \( j \)

Output: Feasible and Meaningful solutions

1. Initialize \( C_{\text{Input}}, C_{\text{Boundary}(i, j)}, C_{\text{Meaningful}(i)} \);
2. \( C_{\text{Checked}} \leftarrow \emptyset \);
3. \( \text{Solutions} \leftarrow \emptyset \);
4. For all valid AP \( A \) do
   5. \( s \leftarrow \text{new Z3 solver} \);
   6. \( s.add([C_{\text{Input}}, C_{\text{Boundary}(i, j)}, C_{\text{Meaningful}(i)}]) \);
   7. \( s.add(\neg C_{\text{Checked}}) \);
   8. \( s.add([C_{A}(A), C_{\text{Forward}}(A)]) \);
   9. If \( s.solve() == \text{satisfiable} \) then
      10. \( \text{Solutions}.\text{Append}(s.\text{model}()) \);
   11. \( C_{\text{Checked}} \leftarrow C_{\text{Checked}} \lor C_{A}(A) \);
5. Return \( \text{Solutions} \);

To find samples \( y \) that belong to class \( i \) and are on the decision boundary of class \( j \) (i.e., \( F(y)_i = F(y)_j \geq F(y)_{\forall k \neq i, j} \)), we need to enumerate all target-boundary class pairs \((i, j)\), which form a complete set of samples supporting the ADRs and are referred to as boundary samples.

Algorithm 1 presents a complete workflow for this implementation. Line 1 initializes the input, boundary, and meaningful constraints, which are shared for each valid activation pattern. In line 2, we use \( C_{\text{checked}} \) to track the regions that have already been checked in previous activation patterns to avoid redundancy, as described in Section IV-A. The \( \text{Solutions} \) list in line 3 stores the solved feasible and meaningful samples. In lines 4-8, we create a Z3 solver \( s \) for each valid activation pattern \( A \) and add the required constraints to it. If the constraints can be solved, we append the generated sample to \( \text{Solutions} \) as shown in lines 9-11. A checked region is added to \( C_{\text{checked}} \) in line 12 to avoid solving it again for other activation patterns. The algorithm returns all feasible and meaningful solutions for target class \( i \) and boundary class \( j \).

We now discuss the details for implementing enumeration of activation patterns in line 4 of Algorithm 1. Recall that there are \( \frac{m}{2} \) possible values for \( \text{Act}_i \) and \( \text{Ina}_i \), respectively. The unique constraint on \( \text{Act}_i \) and \( \text{Ina}_i \) is \( (\text{Act}_i \neq \text{Ina}_i) \lor \text{Act}_i \neq \frac{m}{2} \), since any group cannot be both active and inactive door, except one case that \( \text{Act}_i = \text{Ina}_i = \frac{m}{2} \), i.e., the groups are neither activated nor inactivated. We arrange all activation patterns in a tree structure. In this way, we can implement the enumeration of activation patterns by breadth-first searching and execute from the shallow layers to the deep layers.

The experiment includes two parts: the utilization of autoencoders and the generation of boundary and adversarial examples. Autoencoders are employed to generate prototypes for each dataset that represent meaningful samples with explicit semantics. The prototypes can be used for global verification, distinct from instance-wise local robustness. Boundary samples were produced for each class by identifying samples that are situated on the decision boundary between that class and the adjacent class. Adversarial examples were generated from both the exact and relaxed boundary regions. Starting from the boundary samples, perturbations were added to create adversarial examples. More details can be found in the references.

VI. CONCLUSION

In this paper, we provide a complete and refined definition of SDNs and ADRs in the DeepGlobal framework. We then present a complete specification of the framework using the SMT solver Z3 and demonstrate its detailed algorithmic implementation. Additionally, we leverage prototypes crafted by autoencoder to improve the verification framework by searching for meaningful solutions. The experiments on two benchmark datasets show that increasing the activation coefficient \( \alpha \) will lead to better model performance. Besides, the proposed specification support the generation of extensive boundary and adversarial samples, which can be used for identifying global ADRs of a given model. The selected customized tactics in Z3 further improve the effectiveness of our framework.

ACKNOWLEDGEMENT

This research was sponsored by NSFC under Grant No. 62172019, and CCF-Huawei Populus Grove Fund.

REFERENCES

Blockchain-based Food Traceability System for Apulian Marketplace: Enhancing Transparency and Accountability in the Food Supply Chain

Marco Fiore  
Dep. of Electrical and Information Eng.  
Polytechnic University of Bari  
Bari, Italy  
marco.fiore@poliba.it

Marina Mongiello  
Dep. of Electrical and Information Eng.  
Polytechnic University of Bari  
Bari, Italy  
marina.mongiello@poliba.it

Giovanni Tricarico  
Confcooperative Puglia  
Viale Einaudi, 15  
Bari, Italy  
tricarico.g@confcooperative.it

Francesco Bozzo  
SINAGRI s.r.l.  
University of Bari Aldo Moro  
Bari, Italy  
francesco.bozzo@uniba.it

Cinzia Montemurro  
Department of Soil, Plant and Food Science  
University of Bari Aldo Moro  
Bari, Italy  
cinzia.montemurro@uniba.it

Alessandro Petrontino  
SINAGRI s.r.l.  
University of Bari Aldo Moro  
Bari, Italy  
ale.petrontino@gmail.com

Clemente Giambattista  
Giardinetto Soc. Cooper.  
S.S. 90 km. 63.300 - B.go Giardinetto  
Orsara di Puglia (FG), Italy  
servizi@giardinetto.net

Giorgio Mercuri  
Giardinetto Soc. Cooper.  
S.S. 90 km. 63.300 - B.go Giardinetto  
Orsara di Puglia (FG), Italy  
servizi@giardinetto.net

Abstract—Traceability is a useful tool for consumers to gather as much information as possible about a particular product. Businesses, on the other hand, see traceability as a strategic marketing tool because it allows them to ensure the quality of their goods to customers in a transparent manner. The ability to readily access all information about an agri-food product is critical to customer trust. Products’ information can include where they were manufactured, where they came from, what steps they took to reach at the shelter, and so on. The Blockchain technology is an illustration of how all industries are shifting toward technology and communication. The aim of this paper is to present the Tracecoop project and give an overview of the architecture of the proposed system. The platform ensures trust and guarantees a sense of community both for the consumer and the producer.

Index Terms—Blockchain, traceability, agri-food, supply chain, transparency

I. INTRODUCTION

A Blockchain-based traceability method can boost retailers’ impact and trust. Policymakers induced retailers to improve food safety as a result of the incidence of several food scandals in recent years [1]. Blockchain technology enables consumers to track a product’s complete life cycle, reducing food fraud. Trust is an important element in interactions between service providers and customers. Trust is an essential driver of patronage intentions in the context of stationary retailing. Convenience, origin, and fairness are the most significant elements influencing consumer decisions [2]. Improved traceability allows for the rapid identification of the point of origin in the event of food poisoning. Of course, food traceability is useful if the data saved in the platform cannot be tampered with or altered [3]–[5]. Blockchain’s immutability of data enables it to solve the majority of these problems. Blockchain technologies are trust-proof systems in which non-trusting members can interact with each other in a verifiable manner without the need for a trusted third-party authority.

Blockchain-based traceability systems are currently an interesting topic for researchers [6]–[8]. Authors of paper [9] focus their attention on the need of an agri-food traceability system in supply chains. They develop a system that takes advantage of Radio-Frequency Identification (RFID) devices and use them to track products, notarizing every action in the Blockchain. Such system can enhance the agri-food products safety and quality, but some drawbacks regard the cost of maintaining a Blockchain-based system. The same issue is analyzed in paper [10]. Authors present a Blockchain-based traceability solution for the management of the agri-food sector, using Internet of Things (IoT) devices to gather information from the fields. One of the goals of this research is to compare two different Blockchain, Ethereum and Hyperledger Sawtooth, to understand the pros and cons of each of them. The comparison is focused on
computational costs and performance. Sawtooth results in having better performance in terms of CPU load, network traffic and latency. Other researchers also propose different Blockchain systems, such as the ones based on Hyperledger Fabric [11], [12]. To the best of our knowledge, all proposals lack on a complete evaluation of a supply chain, from the cultivation phase to the selling phase. Moreover, there are no solutions based on using context-aware smart contracts, that can adapt to different scenarios and use cases without the need to modify the built system.

The project’s aim is to create a prototype of the TRACE-COOP system by implementing a Blockchain based on the Hyperledger Fabric and Ethereum frameworks to handle information about agricultural supply chain production processes. The tool will provide the final user with instant and comprehensive access to all of the product’s properties and characteristics, such as its origin, cultivation, use of chemical substances, organoleptic and nutritional molecular characteristics.

The rest of the paper is organized as follows. Section II gives an overview on Blockchain technology; Section III describes the architecture, Section IV evaluates first results. Finally, Section V draws conclusions.

II. BLOCKCHAIN

Blockchain is a distributed ledger system that allows for the safe and transparent recording of transactions. It is made up of a network of computers that share a database of events that are organized into blocks and linked together in a chain-like structure. Because each block includes a cryptographic hash of the previous block, it is nearly impossible to change previous transactions without also changing subsequent blocks.

There are three types of Blockchain, named public Blockchain, private Blockchain and consortium or federated Blockchain [13].

- Public Blockchain: there are no nodes that take control of the network; anyone can join the distributed ledger and input or access the information stored in it. A public Blockchain is open and transparent; consensus methods include Proof-of-Work (PoW), Proof-of-Stake (PoS), and Proof-of-Authority (PoA).
- Private Blockchain: managed by a company or individual who gets control of the network. Mining rights can be granted to anyone, but the organization makes the final choice. In this situation, the ledger can be considered more centralized than the public Blockchain because a single entity has more rights than others. A private Blockchain costs less than a public one.
- Consortium Blockchain: the owners of the ledger are multiple nodes. The Blockchain becomes more decentralized than the private one, but not as costly as the public one.

The main Blockchain features are proposed below, to better understand its peculiarities.

- Secure: The Blockchain employs cryptographic techniques to ensure transaction security and integrity.
- Immutable: Once a transaction is logged on the Blockchain, it cannot be changed or deleted without network consensus.
- Distributed: The Blockchain is a decentralized system that uses a network of computers to keep a shared transaction database.
- Trustless: Without the need for intermediaries or trust in a central authority, the Blockchain enables parties to transact with one another.
- Transparent: All transactions on the Blockchain are visible to all network participants.
- Interoperable: Blockchain networks can communicate with one another, allowing value to be transferred between systems.
- Consensus-driven: Changes to the Blockchain must be validated and agreed upon by network members using a consensus method such as proof of work or proof of stake.
- Programmable: Smart contracts enable programmable transactions to be executed automatically when certain conditions on the Blockchain are met.

III. ARCHITECTURE DESCRIPTION

The architecture of the system proposed in the project is shown in Fig. 1. It is composed of four main components: the front-end application for both the consumer and the producer, the private Blockchain for sensors, and the back-end application.

- Front-end for the consumer. The front-end component is a web-based application. This app allows the consumer to scan a QR code placed on the products label to access the back-end and read information about the chosen product. The main goal is to guarantee to the consumer that the information shown in the platform are secure and transparent.
- Front-end for the producer. The front-end component for the producer lets various actors (i.e., the farmer, the distributor, the retailer) add information not gathered by IoT sensors. In this way, producers can establish a trust relationship with the consumer; the more the consumer trusts a producer, the more he buys that producer products.
- Private back-end. Data coming from IoT sensors are not directly inserted in the main back-end [14]; instead, they are gathered in a private local one that is responsible of the collection and conversion of information into value. An edge computing unit reads the information coming from sensors and, after a defined amount of time, extracts valuable information and uploads them into the main back-end [15].
- Back-end. This component is in charge of storing information about agri-food products in a safe and immutable way. Data stored in this component are accessible (with readonly permission) from customers who scan the QR code placed on the products label.
The proposed architecture is the result of the evaluation of some relevant issues:

- The system must be available 24h, being it used in different periods of the day.
- Sensor’s data are constantly generated, so the system must support huge amounts of information.
- The system must be scalable and optimized for different kinds of products and not only for the test ones.
- There must be the possibility to add new functions without turning the platform off.

To satisfy the requirements described above, we decided to use a Blockchain-based architecture. Blockchain makes the history of any digital asset unalterable and transparent through the use of decentralization and cryptographic hashing. A Blockchain is a distributed database, shared among network nodes. It stores information electronically in digital representation as a database. Blockchains are best known for their critical position in cryptocurrency systems like Bitcoin, where they keep a secure and decentralized record of transactions. The Blockchain’s innovation is that it ensures the fidelity and security of a data record and creates trust without the need for a trusted third party.

### IV. RESULTS EVALUATION

The performed evaluation has a dual objective, on the producer side and on the consumer side. **Producer side**, the goal is to provide an advantage in terms of economic and organizational benefits: the goal is a) to evaluate the increase in added value, b) to improve the market perception with respect to the Blockchain, and c) to evaluate the increase in company performance. **Consumer side**, a survey has been carried out to allow us to assign a monetary value to the consumer’s confidence in the product he is buying, as well as to verify the consumer’s propensity to purchase a product if it is tracked using the Blockchain technology.

The applied method allows us to determine the value of the utility deriving from the purchase of an asset through the preferences of individuals regarding the attributes that characterize it. It was decided to set up the work to follow on consumer preferences, focusing our attention on attributes consistent with the technological aspect of the use of the Blockchain and with the information aspects linked to different areas of sustainability. Blockchain technology, in addition to representing an innovative traceability system, can also be understood as an innovative system for communicating with the consumer.

The attributes to identify individuals preferences are:

- Blockchain technology and QR code on the product label (present, absent).
- Information on environmental sustainability (none, partial, complete).
- Information on social sustainability (none, partial, complete).
- Information on products quality (none, partial, complete).
- Information on company innovation (none, partial, complete).
• Price (about half the average price, average price, 4 times the base price, 6 times the base price).

We refer to Choice Experiments (CE) technique [16], [17], that defines two metrics: Utility Function and Willingness To Pay (WTP).

From the point of view of the econometric analysis of data, it is assumed from the theory of consumer behavior that the latter in choosing between two goods will select the one with greater utility. The utility function is described by a deterministic component $V$, a function of the observable attributes, and a stochastic component $\epsilon$ which represents the measurement errors and all the unobservable attributes that influence the purchase decision:

$$U_{n_j} = V_{n_j} + \epsilon_{n_j}$$

$$V_{n_j} = \beta x_{n_j} = \alpha + \beta_1 x_{1n} + \beta_2 x_{2n} + ... + \beta_m x_{mn}$$

where $n$ denotes the interviewee and $j$ a chosen alternative; with $x_{n_j}$ we indicate the attribute $x$ of the alternative $j$ evaluated by the individual $n$; $\beta$ is the weight of the preference for each level of attribute, as well as the compromises in the monetary value, while the coefficient $\alpha$ incorporates the heterogeneity of the sample.

WTP is calculated as the maximum price an individual is willing to pay for a given attribute or characteristic, based on the choices they made during the experiment. In this way, the utility function can be used to calculate the WTP and thus estimate the value that individuals attribute to the different characteristics of the proposed alternatives. WTP formula is:

$$WTP_a = -\beta_a/\beta_p$$

where $\beta$ represents the specific coefficients estimated for the attribute $a$ and the attribute price $p$.

The survey has reached a provisional number of participants equal to 327. Experiments are taking advantage of focus attribute $\beta$ and respectively. Experiments and validation are being performed with further collection of interviews and processing and evaluation of survey results.

V. CONCLUSION

The TRACECOOP project offers an innovative solution to agri-food traceability, giving producers the possibility to use a Blockchain-based system to ensure data immutability and transparency to customers. The presented project aims at developing a system that is new to market, easy to use both for customers, who do not need to install any application to access the information about the product they are going to buy, and for producers, who can easily gather information from fields using IoT sensors, convert them into valuable information and upload them into a public Blockchain as Ethereum. Some preliminary results show the importance of applying such architecture to the agri-food supply chain: consumers are more likely to trust a product whose traceability is stored in a Blockchain platform. The platform can be extended to be part of a more complete smart city based on Blockchain technology [18]. We are currently performing extensive experiments to provide a comprehensive overview of the market.

ACKNOWLEDGMENT

The authors would like to thank the TRACECOOP project.

REFERENCES


Applying Symbolic Execution to Semantic Code Clone Detection

Kazusa Takemoto  
Dept. of Info. and Comp. Sci., Keio University  
kazusa@doi.ics.keio.ac.jp

Shingo Takada  
Dept. of Info. and Comp. Sci., Keio University  
michigan@ics.keio.ac.jp

Abstract

Many approaches have been proposed to detect code clones, which are basically similar code fragments. Most approaches are based on textual similarity. These approaches cannot detect semantic code clones, which are clones that have the same functionality but implemented with different syntax. Two functions can be considered to have the same functionality, when the output is the same given the same input. In order to appropriately generate inputs, we propose applying symbolic execution to semantic code clone detection. These functions are executed to obtain outputs, which are compared to determine if function pairs are clones. Our approach also does not limit output to return values; we also handle arrays and pointers as output, as the execution of the function may cause changes in their values. Furthermore, we classify types to enable cases where the types of inputs and/or outputs are not exactly the same. We evaluate our approach on SemanticCloneBench.

1. INTRODUCTION

Software developers will make what are called code clones, which are basically similar code fragments. There are four types of code clones, where the first three are focused on syntax [3], and the last one is focused on semantics [8]. Type 1 clone is an exact copy without modifications (except for white space and comments). Type 2 clone is a syntactically identical copy, where only variable, type, or function identifiers are changed. Type 3 clone is copy with further modifications, such as changing, adding, or removing statements. Type 4 clone is two or more code fragments that perform the same computation but are implemented by different syntactic variants. Type 4 code clone is also called semantic code clone, and is the focus of this paper.

The presence of code clones incurs unnecessary costs in maintenance. Manually detecting code clones is time-consuming, so much work has been done on automatically detecting them [1][9]. Many of these approaches compare lexical units or graphically represent the structure of the code and compare the graphs. Such approaches often fail to find semantic code clones. Semantic code clones have also been confirmed in actual software development[5], and need to be addressed.

We focus on the fact that the same input returns the same output when the functions are the same. We detect semantic code clones by comparing the inputs and outputs. In order to generate an appropriate set of inputs, we consider input generation as test case generation and use symbolic execution to generate them. It is also necessary to deal with code clones with different types and to deal with functions other than return values in the comparison. We thus propose an approach for detecting semantic code clones by comparing input/output for each execution path using symbolic execution, taking into account differences in types and outputs having effects other than return values.

Section 2 describes related work and issues. Section 3 describes our proposed approach. Section 4 evaluates our approach, and section 5 makes concluding remarks.

2. RELATED WORK

Several approaches have been proposed for detecting semantic code clones by representing the structure of codes using graphs, e.g., PDG or CFG, and determining whether the code is a semantic code clone or not based on their similarity [1][11][10]. These approaches assume that PDG (or CFG) show the semantics of the code, and do not consider input/output. However, if two functions use a completely different algorithm, their respective PDGs will be different, and thus these approaches cannot find them.

On the other hand, Li et al [6] uses a test-based approach to detect clones in Java. They first filter out APIs that have differing input (parameter) and output (return) types. They further limit their search to methods with similar method names. In addition, the test execution uses branch coverage-based testing to obtain input-output pairs for a single target code, executes each method using its inputs, and compares the outputs of the approaches. The comparison is terminated when test cases with different outputs are found, in which case that method would not be a clone. The issue...
with this approach is that (1) the branch coverage based approach may not obtain all possible execution paths, (2) for each pair of methods under consideration, the approach only executes the test cases derived from the given target method, (3) it does not consider outputs other than the return value, and (4) it cannot detect cases where the parameter and return value types between methods are different. These issues can be summarized as follows:

Issue A: All execution paths are not considered.

Issue B: Test cases are generated only for the given target method and not for the candidate method.

Issue C: Side effects other than explicit outputs (returns) are not considered.

Issue D: Methods with differing parameter types and return types are not considered.

3. PROPOSED APPROACH

3.1. Overview of the proposed approach

We propose an approach where we apply symbolic execution to semantic code clone detection. Our proposed approach targets the C language. Given a set of functions, our approach outputs pairs of semantic code clones.

Our approach first uses symbolic execution to generate test cases (input-output pairs) for each execution path. It then executes each pair of functions against each other with the generated test cases. The output results for each function pair given the same input is checked. The percentage of matching results is given, where 100% indicates that the pair is a semantic code clone. Note that each function pair executes each other’s test cases. In some cases, our approach also compares functions having differing input (parameter) and output (return) types. All test cases are executed and results are compared without terminating, even when test cases executions have different outputs. Because of this, our approach is expected to have a longer execution time compared to Li’s approach [6].

3.2. Features of the proposed approach

Our proposed approach has the following three features to overcome the four issues given in the previous section.

Feature α: To address Issues A and B, our approach compares the execution paths of function pairs to cover the full range of execution paths, thus detecting whether the input/output corresponding to execution path is an exact match or not, as well as similar but not exact match.

Feature β: To address Issue C, if the parameter is an array or pointer, its value is also considered as output. Although the return value is the primary output for a C function, the execution of a function may result in side effects. For example, if a parameter is an array or a pointer, its value may change as a result of operations performed on it within the function. So, if this array or pointer is accessed after this function call, its contents may have been modified, causing a side effect on the program. Thus, we also consider these as output.

Feature γ: To address Issue D, we incorporate type classification to enable the detection of semantic code clones with different input/output types. Two functions may be the same except for the parameter types and return types. We also consider these as semantic code clones. In order to handle the difference in types, we classify types.

3.3. Process Flow

The process flow of our approach is divided into the following five main parts (Figure 1):

1. Information extraction
2. Function grouping
3. Executable file generation for symbolic and test execution
4. Test case generation through symbolic execution and test execution
5. Test execution of each pair and output of match rates

Information extraction  This part extracts and saves the basic information used in subsequent parts. Specifically, signature information of functions (function name, parameter types, and return type), structs (struct name, and each member’s name and type), and typedefs (type name and type) are extracted using the `ctags` command.²

Function grouping  The second part groups the functions using the extracted parameter types and return type. The order of the parameters are not considered in our current implementation. This results in the following two groups:

- Exact match group: Each exact match group contains functions whose return value types, number of parameters, and parameter types exactly matches.
- Type-class match group: Types are classified and grouped to detect semantic code clones with different parameter and return types. We shall call such classified types as type-class. Even if two types are technically different, if they belong to the same type-class, then they will be treated as if they were the same type. Currently, we employ three type-classes as follows:
  - Numeric type-class: int, long, float, etc.
  - Character type-class: char, signed char, unsigned char
  - Set type-class: array, pointer

Note that functions that belong to the same exact match group will also belong to the same type-class match group.

Executable file generation for symbolic and test execution  The information that were extracted in the first part is used to generate executable files for symbolic execution and test execution of each function. The former is used to generate inputs, that will be used by the latter to obtain the outputs. The corresponding input-output pair will be used as test cases. In our current implementation, the length of the area pointed to by arrays and pointers is set to 100.

For symbolic execution, a C function is generated and compiled that takes inputs from standard input, calls the function to be checked, and outputs the execution result to standard output. Note that the inputs are the values that are generated through symbolic execution.

Test case generation through symbolic execution and test execution  The generated executables in the previous part is used to conduct symbolic execution and test execution.

First, symbolic execution is done with the executable generated in the previous part. This results in execution paths which are further analyzed by KLEE resulting in input values corresponding to the execution paths. The number of generated input values is limited to a maximum of 100 for each function to prevent the overall execution time from becoming too long.

Next, test execution is done using the input values that were generated with symbolic execution. For each function, a test run is performed with the generated input values as inputs, and the input-output pairs are saved as test cases. Our current implementation limits the test execution time to two seconds.

Test execution of each pair and output of match rates  Tests are now conducted on functions that are to be checked for the possibility of being semantic code clones. Specifically, all pairs within groups (exact match group and type-class match group) are checked. Note that the check is done two-ways. For example, if a developer wants to check if functionA and functionB are clones, the previous part is conducted to obtain testCasesA and testCasesB, respectively. The inputs in testCasesA are given to functionB, and the inputs in testCasesB are given to functionA. The outputs are checked against the corresponding test cases, and the following formula is used to calculate the match rate ($MR$):

$$MR[\%] = \frac{\text{Number of test cases with matching output}}{\text{Number of test cases used for comparison}} \times 100$$ (1)

A semantic code clone should have a 100% match rate.

4. EVALUATION

4.1. Research Questions

We pose the following three research questions:

- RQ1: How well can semantic code clones be detected?
- RQ2: Can outputs other than return values be considered?

²http://ctags.sourceforge.net
- RQ3: Can our approach support semantic code clones of different input/output types?

4.2. Evaluation Dataset

The evaluation uses SemanticCloneBench [2]. This is a benchmark consisting of semantic code clones from code that exists on Stack Overflow, and was collected based on human judgment. For C language, it contains 1000 pairs of semantic code clones. Since it includes pairs that are out of scope of our tool, we extracted 91 pairs for our evaluation based on a set of conditions. Table 1 shows the conditions, as well as the number of pairs removed due to not satisfying our conditions.

The vast majority of the functions in SemanticCloneBench were main functions, which our approach does not handle. This is because code clone search is normally done within a project, which will likely not have multiple cloned main functions.

<table>
<thead>
<tr>
<th>Conditions of target pairs</th>
<th>Number of deleted pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not main function</td>
<td>720 pairs</td>
</tr>
<tr>
<td>Input/output is parameter/return value</td>
<td>47 pairs</td>
</tr>
<tr>
<td>Struct does not have a circular structure</td>
<td>34 pairs</td>
</tr>
<tr>
<td>Function is executable</td>
<td>76 pairs</td>
</tr>
<tr>
<td>Parameter type is supported by our tool</td>
<td>32 pairs</td>
</tr>
</tbody>
</table>

4.3. Result

Of the 91 pairs, 72 pairs were type matched by function grouping. Of the 72 pairs, one pair could not be checked as execution of test cases ended with errors or the execution time exceeded the limit of two seconds for each test case. Table 2 shows the evaluation results for the remaining 71 pairs, where MR denotes the match rate. The total execution time of the tool was 1353 seconds.

<table>
<thead>
<tr>
<th>Match rate</th>
<th>RQ1</th>
<th>RQ2</th>
<th>RQ3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR = 100%</td>
<td>33</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>80% ≤ MR &lt; 100%</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>50% ≤ MR &lt; 80%</td>
<td>12</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>30% ≤ MR &lt; 50%</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>MR &lt; 30%</td>
<td>16</td>
<td>12</td>
<td>6</td>
</tr>
</tbody>
</table>

4.4. RQ1: How well can semantic code clones be detected?

Column “RQ1” in Table 2 shows the distribution of the 71 pairs in terms of match rate. 33 pairs out of 71 pairs had a match rate of 100%, and can be considered as semantic code clones. As described in section 4.2, SemanticCloneBench [2] is a benchmark containing pairs of functions that were considered to be clones based on human judgment. We expected that the number of pairs with a 100% match rate would be the majority. However, this was not the case. We further manually analyzed the results to understand what caused the differences.

We first manually checked the results when the match rate was 100%. We found that 6 of the 33 pairs were actually not completely the same, i.e., the match rate should not have been 100%. One example is a pair of functions that perform Caesar cipher encryption. One function performs Caesar cipher encryption for the letters A to Z, while the other function performs Caesar cipher encryption for the numbers 0 to 9 in addition to the letters A to Z. Therefore, this pair can be considered to be similar but is partially different, and the match rate should not have been 100%. The reason this occurred was because one of the parameters did not affect the execution path, and thus only one value was generated for that parameter. Multiple values needed to have been generated for that parameter for the difference between the two functions to appear. This can be considered as an issue with using symbolic execution. We are currently considering this as part of future work.

For pairs having a match rate between 80% and 100%, a common issue were cases that had “special” input values. One example was a pair of functions that returned a Fibonacci number. Both functions have the same output value for inputs greater than or equal to 2. But the output was different when the input was less than or equal to 1.

For pairs having a match rate between 50% and 80%, we found cases where the function itself is the same, but the range of input values it can handle differs. One example was where one function supports up to 32-bit input values while the other supports only 16-bit values.

For pairs having a match rate between 30% and 50%, we found cases where one function would execute correctly under specific conditions. In one pair, one of the function would execute correctly even if a string had two or more spaces, but the other would not.

Note that all of these pairs were considered to be semantic code clones as they came from SemanticCloneBench. Furthermore, there are several definitions of semantic code clones, where the core is that (1) the clones have the same functionality, and (2) are implemented with different syntax [2]. All of these pairs had the same basic function, but many had differences, which could be due to exceptional values, input range, or conditions. 100% match rate is a very strict application of “same functionality”, and our results strongly suggests that the definition of semantic code clone may need to be refined depending on the scenario. At the same time, as our approach outputs match rate, the broadening of the semantic scope may be easily attained.
4.5. RQ2: Can outputs other than return values be considered?

Our approach considers outputs that do not take the form of return values. Specifically, our approach can also handle arrays and pointers as outputs if they appear as a parameter. Column “RQ2” in Table 2 shows the number of cases where the output included a parameter.

In some cases, although the array or pointer was handled correctly, the function pairs had differing return values causing the match rate to be low. One such example was a pair of functions that convert numeric values to strings. The two functions were implemented recursively, returning the result of the execution as a return value. The functions assumed that the parameter values (strings) were to be used as the “actual” output, and not necessarily the return value itself. Even if the resulting parameter values were the same, the return values were not always the same. This resulted in the match rate to be low. In order to deal with such cases, we need to take into consideration such factors as the partial match of outputs.

4.6. RQ3: Can our approach support semantic code clones of different input/output types?

Our approach classified types into type-class to enable finding semantic code clones even if the types do not exactly match. Column “RQ3” in Table 2 shows the number of cases where the input and/or the output types differed between the two functions, showing the importance of being able to handle such cases.

4.7. Threats to validity

First, the number of target pairs used in the evaluation is a threat. SemanticCloneBench has 1000 pairs of functions. However, most of them were main functions or used standard input/output, which our approach does not handle. In order to strengthen the generality of our evaluation, it is necessary to conduct evaluation on a larger data set.

Second, the execution time of symbolic execution, the execution time of tests, the length of the area pointed to by arrays and pointers, and the number of test cases generated for each function were all fixed. The experimental results may differ if these parameter values are changed.

Third, there is the possibility of human error when manual analysis was done in RQ1. We took great care to limit this possibility.

5. CONCLUSION AND FUTURE WORK

We proposed an approach that used symbolic execution to detect semantic code clones. The novel part of our approach is (1) the comparison of the execution paths of function pairs to cover the full range of execution paths, (2) the ability to handle outputs other than return values, specifically pointers and arrays, and (3) type classification to enable cases where input/output types are not exactly the same. Evaluation using SemanticCloneBench showed the viability of our approach.

Future work includes the following. First, we need to consider the definition of semantic code clone. Our definition is currently strict, i.e., the input/output needs to match 100% between functions. Second, we need to consider cases where the number of parameters differs. Third, we need to consider cases where a parameter does not affect the execution path, but will affect the variety of inputs needed to more correctly obtain the match rate. Finally, we need to consider other types of outputs, e.g., global variables and print functions such as printf().

References

Session ESSE: Empirical Studies and Software Engineering
Practices and Challenges of Using GitHub Copilot: An Empirical Study

Beiqi Zhang†*, Peng Liang††*, Xiyu Zhou††, Aakash Ahmad§, Muhammad Waseem††
†School of Computer Science, Wuhan University, Wuhan, China
‡Hubei Luojia Laboratory, Wuhan, China
§School of Computing and Communications, Lancaster University Leipzig, Leipzig, Germany
{zhangbeiqi, lianggp, xiyzhou, m.waseem}@whu.edu.cn, a.ahmad13@lancaster.ac.uk

Abstract—With the advances in machine learning, there is a growing interest in AI-enabled tools for autocompleting source code. GitHub Copilot, also referred to as the “AI Pair Programmer”, has been trained on billions of lines of open source GitHub code and is one of such tools that has been increasingly used since its launch in June 2021. However, little effort has been devoted to understanding the practices and challenges of using Copilot in programming with auto-completed source code. To this end, we conducted an empirical study by collecting and analyzing the data from Stack Overflow (SO) and GitHub Discussions. More specifically, we searched and manually collected 169 SO posts and 655 GitHub discussions related to the usage of Copilot. We identified the programming languages, IDEs, technologies used with Copilot, functions implemented, benefits, limitations, and challenges when using Copilot. The results show that when practitioners use Copilot: (1) The major programming languages used with Copilot are JavaScript and Python, (2) the main IDE used with Copilot is Visual Studio Code, (3) the most common used technology with Copilot is Node.js, (4) the leading function implemented by Copilot is data processing, (5) the significant benefit of using Copilot is useful code generation, and (6) the main limitation encountered by practitioners when using Copilot is difficulty of integration. Our results suggest that using Copilot is like a double-edged sword, which requires developers to carefully consider various aspects when deciding whether or not to use it. Our study provides empirically grounded foundations and basis for future research on the role of Copilot as an AI pair programmer in software development.

Keywords—GitHub Copilot, Stack Overflow, GitHub Discussions, Empirical Study

I. INTRODUCTION

Large Language Models (LLMs) and Machine Learning (ML) for autocompleting source code are becoming more and more popular in software development. LLMs nowadays incorporate powerful capabilities for Natural Language Processing (NLP) [1], and ML approaches have been widely applied to source code text in a variety of new tools to support software development [2], which makes it possible to use LLMs to synthesize code in general-purpose languages [1]. Recently, NLP-based code generation tools have come into the limelight, with generative pre-trained language models trained on large amounts of code in an attempt to provide reasonable auto-completion of the source code when programmers write code [5]. Released on June 2021, GitHub Copilot has recently emerged as an “AI pair programmer”, which is powered by OpenAI Codex and suggests code or entire functions in IDEs as a plug-in [4] to help developers achieve code auto-completion in programming activities.

Although the emergence of AI-assisted programming tools has empowered practitioners in their software development efforts, there is little evidence and lack of empirically-rooted studies (e.g., [3], [5], [6]) on the role of AI-assisted programming tools in software development. The existing studies primarily focus on the correctness and understanding of the code suggested by Copilot, and little is known about the practices and challenges of using Copilot with programming activities.

To close the gap, we conducted this study that collects data from Stack Overflow (SO) and GitHub Discussions to get practitioners’ perspectives on using Copilot during software engineering and development.

The contributions of this work: (1) we identified the programming languages, IDEs, and technologies used with Copilot; (2) we provided the functions implemented by Copilot, the benefits, limitations, and challenges of using Copilot; and (3) we present the directions to be further explored.

II. RELATED WORK

Several studies focused on the security issues of Copilot. Sandoval et al. [7] conducted a user study to investigate the impact of programming with LLMs that support Copilot. Their results show that LLMs have a positive impact on the correctness of functions, and they did not find any decisive impact on the correctness of safety. Several studies focused on the quality of the code generated by Copilot. Imai [8] compared the effectiveness of programming with Copilot versus human programming, and found that the generated code by Copilot is inferior than human-written code. Yetistiren et al. [9] assessed the quality of generated code by Copilot in terms of validity, correctness, and efficiency. Their empirical analysis shows Copilot is a promising tool. Madi et al. [6] focused on readability and visual inspection of Copilot generated code. Through a human experiment, their study highlights that programmers should beware of the code generated by tools. Wang et al. [10] collected practitioners’ expectations on code generation tools through a mixed-methods approach.
They found that effectiveness and code quality is more important than other expectations. Several studies focused on the limitations and challenges in Copilot assisted programming. Dakhel et al. [11] explored the capabilities of Copilot through empirical evaluations, and their results suggest that Copilot shows limitations as an assistant for developers. Nguyen and Nadi [12] conducted an empirical study to evaluate the correctness and comprehensibility of the code suggested by Copilot. Their findings revealed that Copilot’s suggestions for different programming languages do not differ significantly, and they identified potential shortcomings of Copilot, like generating complex code. Bird et al. [6] conducted three studies to understand how developers use Copilot and their findings indicated that developers spent more time assessing Copilot’s suggestions than doing the task by themselves. Sarkar et al. [13] compared programming with Copilot to previous conceptualizations of programmer assistance to examine their similarities and differences, and discussed the issues that might arise in applying LLMs to programming.

Compared to the existing work (e.g., [9], [11]), our work intends to understand the practices and challenges of Copilot by exploring the programming languages, IDEs, technologies used with Copilot, functions implemented by Copilot, and the benefits, limitations, and challenges of using Copilot.

III. RESEARCH DESIGN

A. Research Questions

The Research Questions (RQs) and their rationale are presented in Table I. The overview of the research process is shown in Figure 1 and detailed below.

![Fig. 1: Overview of the research process](image)

B. Data Collection and Filtering

This study focuses on the practices and challenges of using Copilot collected from SO and GitHub Discussions. SO is a popular software development community and has been widely used by developers to ask and answer questions as a Q&A platform. GitHub Discussions is a feature of GitHub used to support the communication among the members of a project. Different from SO, GitHub Discussions can provide various communication intentions, not just question-answering (e.g., a discussion can report errors or discuss the potential development of a software project) [14], from which the data can be complementary to the data from SO. Besides, GitHub Discussions can provide a center of a community knowledge base connected with other artifacts in a project [14]. Therefore, we decided to use SO and GitHub Discussions as the data sources to answer our RQs, and we conducted the searches for both SO and GitHub Discussions on November 23rd, 2022.

For SO, “copilot” is used as the term to search the posts related to Copilot. After searching, we got a total of 557 posts that include the search term “copilot”. The term “copilot” may appears more than once in a post, so there may be duplicates in the URL collection of these retrieved posts. After removing the duplicates, we ended up with 521 posts with unique URLs. To manually label posts related to Copilot, we conducted a pilot data labelling by two authors with 10 retrieved SO posts. Specifically, the inclusion criterion is that the post must provide information referring to Copilot. We calculated the Cohen’s Kappa coefficient [15] to measure the consistency of labelled posts, which is 0.773, thus indicating a decent agreement between the two coders. After excluding the irrelevant posts in the search results, we finally got 169 Copilot related SO posts.

For GitHub Discussions, GitHub discussions are organized according to categories. After exploring the categories on GitHub Discussions, we found the “Copilot” category which contains the feedback, questions, and conversations about Copilot [16] under the “GitHub product categories”. Since all the discussions under the “Copilot” category are related to Copilot, we then included all the discussions under the “Copilot” category as related discussions to extract data. The number of discussions related to Copilot is 655.

---

### TABLE I: Research Questions and their Rationales

<table>
<thead>
<tr>
<th>Research Question</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RQ1</strong>: What programming languages are used with GitHub Copilot?</td>
<td>Copilot can help practitioners write less code. This RQ aims to collect the programming languages that developers tend to use with Copilot.</td>
</tr>
<tr>
<td><strong>RQ2</strong>: What IDEs are used with GitHub Copilot?</td>
<td>Copilot is a third-party plug-in used in IDEs. This RQ aims to identify the IDEs frequently used with Copilot. The answers of this RQ can help developers choose which IDE to use when they code with Copilot.</td>
</tr>
<tr>
<td><strong>RQ3</strong>: What technologies are used with GitHub Copilot?</td>
<td>When writing code, programmers need to employ certain technologies to complete the development. This RQ aims to investigate the technologies that can be used with Copilot (e.g., frameworks), and the answers of this RQ can help developers choose the technologies when they use Copilot.</td>
</tr>
<tr>
<td><strong>RQ4</strong>: What functions are implemented by using GitHub Copilot?</td>
<td>Copilot can complete entire functions according to users’ comments. This RQ aims to provide a categorization of the functions that can be implemented by Copilot, and the answers of this RQ can provide developers guidance when implementing functions by using Copilot.</td>
</tr>
<tr>
<td><strong>RQ5</strong>: What are the benefits of using GitHub Copilot?</td>
<td>Using Copilot to assist programming can bring many benefits (e.g., reducing the workload of developers). This RQ aims to collect the advantages brought to the development by applying Copilot.</td>
</tr>
<tr>
<td><strong>RQ6</strong>: What are the limitations and challenges of using GitHub Copilot?</td>
<td>Although using Copilot to assist in writing code can help developers with their programming activities, there are still restrictions and problems when using Copilot. This RQ aims to collect and identify the limitations and challenges practitioners may experience when using Copilot. The answers of this RQ can help practitioners make an informed decision when deciding whether to code with the help of Copilot.</td>
</tr>
</tbody>
</table>
C. Data Extraction and Analysis

1) Extract Data: To answer the RQs in Section III-A, we extracted the data items listed in Table III. The first and third authors conducted a pilot data extraction independently with 10 SO posts and 10 GitHub discussions randomly selected from the 169 SO posts and 655 GitHub discussions. The second author was involved to discuss with the two authors and came to an agreement if any disagreements were found during the pilot. After the pilot, the criteria for data extraction were determined: (1) for all the data items listed in Table III they will be extracted and counted only if they are explicitly mentioned by developers that they were used with Copilot; (2) if the same developer repeatedly mentioned the same data item in an SO post or a GitHub discussion, we only counted it once. In a post or discussion, multiple developers may mention Copilot related data items, resulting in situation that the total number of instances of certain data item extracted may be greater than the number of posts and discussions. The first and third authors further extracted the data items from the filtered posts and discussions according to the extraction criteria, marked uncertain parts, and discussed with the second author to reach a consensus. Finally, the first author rechecked all the extraction results by the two authors from the filtered posts and discussions to ensure the correctness of the extracted data.

<table>
<thead>
<tr>
<th>#</th>
<th>Data Item</th>
<th>Description</th>
<th>Analysis Method</th>
<th>RQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Programming language</td>
<td>Programming language used with Copilot</td>
<td>Descriptive statistics</td>
<td>[17]</td>
</tr>
<tr>
<td>D2</td>
<td>IDE</td>
<td>IDEs used with Copilot</td>
<td>Descriptive statistics</td>
<td>[17]</td>
</tr>
<tr>
<td>D3</td>
<td>Technology</td>
<td>Technologies used with Copilot</td>
<td>Descriptive statistics</td>
<td>[17]</td>
</tr>
<tr>
<td>D4</td>
<td>Function</td>
<td>Functions implemented by Copilot</td>
<td>Constant comparison</td>
<td>[18]</td>
</tr>
<tr>
<td>D5</td>
<td>Benefit</td>
<td>Benefits brought by using Copilot</td>
<td>Constant comparison</td>
<td>[18]</td>
</tr>
<tr>
<td>D6</td>
<td>Limitation/Challenge</td>
<td>The restrictions and difficulties when using Copilot</td>
<td>Constant comparison</td>
<td>[18]</td>
</tr>
</tbody>
</table>

2) Analyze Data: For RQ1, RQ2, and RQ3, we used descriptive statistics [17] to analyze and present the results. For RQ4, RQ5, and RQ6, we conducted a qualitative data analysis by applying the Constant Comparison method [18]. We constantly compared each part of the data (e.g., emergent codes) to explore differences and similarities in the extracted data to form categories [19]. Note that for answering RQ4, we categorized the functions (D4) based on developers’ discussions, i.e., developers’ descriptions of the mentioned functions. Firstly, the first and the third authors coded the filtered posts and discussions with the corresponding data items listed (see Table III). Secondly, the first author reviewed the coded data by the third author to make sure the extracted data were coded correctly. Finally, the first author combined all the codes into higher-level concepts and turned them into categories. After that, the second author examined the coding and categorization results, in which any divergence was discussed till the three authors reached an agreement. The data analysis methods with their corresponding data items and RQs are listed in Table III. The data analysis results are provided in [20].

IV. RESULTS

This section presents the results of the study, in which the results of RQ1 to RQ4 are visualized in Fig 2 and the results of RQ5 and RQ6 are provided in Table III and IV.

RQ1: Programming languages used with GitHub Copilot

Figure 2a lists the 18 programming languages used with Copilot, in which JavaScript and Python are the most frequently used ones, both accounting for one fifth. Besides, developers often write C# and Java code when using Copilot, as one practitioner mentioned “the GitHub Copilot extension is enabled in my VS 2022 C# environment” (GitHub #14115). TypeScript, Rust, PHP, C, Golang, and Kotlin were used with Copilot between 3-8 times each (1.7% to 8.4%). The rest of programming languages (e.g., Perl and Ruby), which are not popular, were mentioned only once with Copilot.

RQ2: IDEs used with GitHub Copilot

Figure 2b shows 22 types of IDEs that are used with Copilot. Visual Studio Code is the dominant IDE, accounting for 46.0%. When first released, Copilot only worked with Visual Studio Code editor, and it is expected that Visual Studio Code is the IDE most often used with Copilot. Mainstream code editors, including Visual Studio, IntelliJ IDEA, NeoVim, and PyCharm are also occasionally used, accounting for 39.9% in total. The remaining IDEs were rarely mentioned by developers, and one possible reason is that there are often integration issues when using Copilot within them according to the results of RQ6.

RQ3: Technologies used with GitHub Copilot

Figure 2c presents 22 technologies used with Copilot. We find that these identified technologies include frameworks, APIs, and libraries. Node.js, whose proportion is more than 40%, is one of the most popular back-end runtime environments for JavaScript, which is also the most frequently used language with Copilot (see the results of RQ1), thus it is reasonable that Node.js is the major technology used with Copilot. In addition, .NET which works for Web development, and Vue, React, and Ajax which are frameworks for front-end development, were mentioned less often compared to Node.js. The rest of the identified technologies, many of which relate to machine learning (e.g., Pandas, Dlib, and OpenCV) or front-end development (e.g., Htmx, Vanilla JS, and Next.js), are rarely used with Copilot, and each of them appears only once.

RQ4: Functions implemented by using GitHub Copilot

Figure 2d shows 14 functions implemented by using Copilot, with Copilot, in which each of them appears only once. The main function implemented by Copilot is data processing, indicating that developers tend to make use of Copilot to write functions working with data. Besides, Front-end element control, string processing, and Test account for the same, i.e., 11.1%. When implementing functions, developers also use Copilot to code image processing, algorithm, iteration, calculation, filtering, printing, memory read, serialization, and URL building, which range from 2.2% to 8.9%.
RQ5: Benefits of using GitHub Copilot

Table III highlights 10 benefits of using Copilot. Most developers mentioned that they used Copilot for useful code generation, which reduced their workload and gave them help when they have no idea about how to write code. Programming with Copilot also brings faster development, as one discussion remarked, Copilot “saves developers a lot of time” (GitHub #35850). Meanwhile, better code quality can be obtained by using Copilot. Compared to the code written by developers themselves, the code suggested by Copilot is usually shorter and more correct, as one developer said, “often Copilot is smarter than me” (SO #74512186). Copilot can use machine learning models to learn code style of developers, so as to offer good adaptation to users’ code patterns. Three developers mentioned that Copilot can give them better user experience than other AI-assisted programming tools, for example, one developer stated that “Copilot works totally different compared to all the other products out there, it is a lot more fun to use and does not annoy me like some other AI systems” (GitHub #7254), without providing the names of the other products.

RQ6: Limitations and challenges of using GitHub Copilot

Table IV lists 15 limitations and challenges of using Copilot. Most developers pointed out the difficulty of integration between Copilot and IDEs or other plug-ins. After Copilot was installed in developers’ IDEs, certain plug-ins did not work and Copilot may conflict with some shortcut settings of the editors. Moreover, Copilot cannot be successfully integrated with some IDEs as Copilot does not support these editors yet. Due to the instability of Copilot servers, developers may have difficulties of accessing Copilot. The code suggested by
Copilot has restrictions as well, and sometimes it just offers few solutions, which are not enough for users, which brings limitation to code generation, as one developer said “multiple solution is too little” (GitHub #373034). Practitioners also complained about the poor quality of generated code by Copilot. Some practitioners said that “GitHub Copilot suggest solutions that don’t work” (SO #73701039), and some practitioners found that when the code files became larger, the quality of the code suggested by Copilot “becomes unacceptable” (GitHub #9282). When using Copilot, developers pay much attention to code privacy threat as well. They were worried that Copilot may use their code information without permission. Contrary to the developers who mentioned that Copilot gave them a better user experience than other AI-assisted programming tools, some practitioners said they had an unfriendly user experience when coding with Copilot.

Besides, Copilot may consider improving the integration of Copilot by supporting more IDEs in the future.

**Support for Front-end and Machine Learning Development:** As we can see from the results of RQ1, RQ3, and RQ4, practitioners often write JavaScript and Python code when using Copilot, and they tend to use Copilot with front-end and machine learning related technologies (including frameworks, APIs, and libraries) to implement front-end (e.g., front-end element control) and machine learning functions (e.g., data processing and image processing). JavaScript is the foundation language of many popular front-end frameworks and most of Websites use JavaScript on the client side. Python is the first choice when it comes to the development of machine learning solutions with the help of rich libraries, e.g., OpenCV. It is consequently reasonable that developers tend to use Copilot with JavaScript to facilitate and generate code for front-end and Python for machine learning development.

**Potentials and Perils of Using Copilot in Software Development:** Trained on billions of lines of code, Copilot can turn natural language prompts into coding suggestions across dozens of programming languages and make developers code faster and easier [4]. The results of RQ5 and RQ6 show that many benefits of using Copilot contradict its limitations and challenges, e.g., useful code generation vs. limitation to code generation. When deciding to use Copilot, developers should consider tool integration, user experience, budget, code privacy, and some other aspects, and make trade-offs between these factors. In short, using Copilot is like a double-edged sword, and practitioners need to consider various aspects carefully when deciding whether or not to use it. If Copilot can be used with appropriate programming languages and technologies to implement functions required by users correctly in developers’ IDEs, it will certainly optimize developers’ coding workflow and do what matters most - building software by letting AI do the redundant work. Otherwise, it will bring difficulties and restrictions to development, making developers

<table>
<thead>
<tr>
<th>Limitation &amp; Challenge</th>
<th>Example</th>
<th>Count</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty of integration</td>
<td>Copilot only works with VSCode, VSCode is not supported at the moment (GitHub #14837)</td>
<td>75</td>
<td>28.0%</td>
</tr>
<tr>
<td>Difficulty of accessing Copilot</td>
<td>I cannot connect to the GitHub account and the Copilot server in VSCode, also cannot use the Copilot plugin (SO #74398521)</td>
<td>47</td>
<td>17.5%</td>
</tr>
<tr>
<td>Limitation to code generation</td>
<td>Copilot is limited to around 1000 characters in the response (GitHub #15122)</td>
<td>39</td>
<td>14.6%</td>
</tr>
<tr>
<td>Poor quality of generated code</td>
<td>GitHub Copilot suggest solutions that don’t work (SO #73701039)</td>
<td>31</td>
<td>11.6%</td>
</tr>
<tr>
<td>Code privacy threat</td>
<td>Copilot does collect personal data so just take precaution when working in private repos (GitHub #1163)</td>
<td>20</td>
<td>7.5%</td>
</tr>
<tr>
<td>Unfriendly user experience</td>
<td>I had the same problem today, an amazing tool with poor user experience (GitHub #8468)</td>
<td>14</td>
<td>5.2%</td>
</tr>
<tr>
<td>High pricing</td>
<td>It is obvious that no one in South America will pay that price, it is too expensive (GitHub #23594)</td>
<td>11</td>
<td>4.1%</td>
</tr>
<tr>
<td>Difficulty of understanding the generated code</td>
<td>I really do not understand this enough, and have no idea half of what this code does honestly. It was written by Copilot. (SO #7282605)</td>
<td>9</td>
<td>3.4%</td>
</tr>
<tr>
<td>No edition for organizations</td>
<td>Currently, Copilot is only available for individual user accounts and organizations aren’t able to purchase/manage Copilot for their members just yet (GitHub #32775)</td>
<td>7</td>
<td>2.7%</td>
</tr>
<tr>
<td>Lack of customization</td>
<td>My question is about setting up shortcuts in Visual Studio Code. (GitHub for GitHub Copilot Labs. (SO #73564811)</td>
<td>5</td>
<td>1.9%</td>
</tr>
<tr>
<td>Difficulty of subscription</td>
<td>My copilot subscription suddenly stopped. Tried log out and in. Never have reply on support ticket over 10 days (GitHub #36190)</td>
<td>3</td>
<td>1.1%</td>
</tr>
<tr>
<td>Challenge of not providing outdated suggestions</td>
<td>making sure that the tool does not provide outdated suggestions would still be a challenge (SO #7254382)</td>
<td>2</td>
<td>0.7%</td>
</tr>
<tr>
<td>Show loading</td>
<td>I am not sure what is causing this but while editing files within Visual Studio, I am periodically locking up with the following dialog showing (SO #73682137)</td>
<td>2</td>
<td>0.7%</td>
</tr>
<tr>
<td>Hard to configure</td>
<td>Keep getting “Your Copilot experience is not fully configured, complete your setup” in Visual Studio 2022 (GitHub #19556)</td>
<td>2</td>
<td>0.7%</td>
</tr>
<tr>
<td>Need of basic programming knowledge</td>
<td>It is useless if you do not understand the programming language or the task you want to do (GitHub #35850)</td>
<td>1</td>
<td>0.4%</td>
</tr>
</tbody>
</table>
feel frustrated and constrained. The study results can help practitioners being aware of the potential advantages and disadvantages of using Copilot and thus make an informed decision whether to use it for programming activities.

Towards an Effective Use of Copilot: Further investigation about the practices of Copilot can be conducted by questionnaire and interview. Under what conditions the challenges of using Copilot will show up as advantages or disadvantages, and how to use Copilot to convert its disadvantages into advantages are also worth further exploration. Besides, although we have investigated various aspects of using Copilot (e.g., limitations and challenges), we have not looked in depth at what types of users (e.g., developers, educators, and students) who use Copilot, when and how they use Copilot, and for what specific purposes. By exploring these aspects, researchers can get meaningful information which would help guide towards an effective use of Copilot.

VI. Threats to Validity

Construct validity: We conducted data labelling, extraction, and analysis manually, which may lead to personal bias. To reduce this threat, the data labelling of SO posts was performed after the pilot labelling to reach an agreement between the authors. The data extraction and analysis was also conducted by two authors, and the first author rechecked all the results produced by the third author. During the whole process, the first author continuously consulted with the second author to ensure there are no divergences.

External validity: We chose two popular developer communities (SO and GitHub Discussions) because SO has been widely used in software engineering studies and GitHub Discussions is a new feature of GitHub for discussing specific topics [13]. These two data sources can partially alleviate the threat to external validity. However, we admit that our selected data sources may not be representative enough to understand all the practices and challenges of using Copilot.

Reliability: We conducted a pilot labelling before the formal labelling of SO posts with two authors, and the Cohen’s Kappa coefficient is 0.773, indicating a decent consistency. We acknowledge that this threat might still exist due to the small number of posts used in the pilot. All the steps in our study, including manual labelling, extraction, and analysis of data were conducted by three authors. During the process, the three authors discussed the results until there was no any disagreements in order to produce consistent results. In addition, the dataset of this study that contains all the extracted data and labelling results from the SO posts and GitHub Discussions has been provided online for validation and replication purposes [20].

VII. Conclusions

We conducted an empirical study on SO and GitHub Discussions to understand the practices and challenges of using GitHub Copilot from the practitioners’ perspective. We used “copilot” as the search term to collect data from SO and collected all the discussions under the “Copilot” category in GitHub Discussions. Finally, we got 169 SO posts and 655 GitHub discussions related to Copilot. Our results identified the programming languages, IDEs, technologies used with Copilot, functions implemented by Copilot, and the benefits, limitations, and challenges of using Copilot, which are first-hand information for developers.

In the next step, we plan to further explore when to use Copilot, for what specific purposes, and by whom, which helps to guide towards an effective use of Copilot.

REFERENCES

An Empirical Study of Coding Style Compliance on Stack Overflow

Qing Mi*, Haotian Bai, Xiaozhou Wang, Wenrui Liu, Xingyue Song
Faculty of Information Technology, Beijing University of Technology, Beijing, China
miqing@bjut.edu.cn, baihaotian20020905@emails.bjut.edu.cn, Wang-xz@emails.bjut.edu.cn,
wenrui.liu@ucdconnect.ie, sxy18386137856@163.com

Abstract—Stack Overflow (SO) is one of the world’s largest technical Q&A websites, in which many posts contain code snippets. However, these code snippets may not comply with coding style guidelines and result in the problem of low readability and maintainability. To provide a better understanding of this coding style compliance issue for SO users, we plan and conduct an empirical study on SO. Specifically, we collected over 400,000 code snippets from SO in three languages, namely Python, C/C++, and JavaScript. The posts are divided into two types (i.e., question and answer) and analyzed separately. We found that for the question- and answer-type posts, more than 90% and 60% of code snippets contain style violations. The most frequently found violation is syntax errors for Python and indentations for C/C++ and JavaScript. In addition, the results show that with more violations in code snippets, the “Score” of Python and C/C++ posts, the “FavoriteCount” of C/C++ questions, and the “CommentCount” of JavaScript questions tend to be lower. The findings of our research indicate that code snippets on SO do not have good coding style compliance. Users, especially programming beginners are supposed to be wary of the potential problems of reusing code snippets on SO.

Index Terms—Coding Style Compliance, Stack Overflow, Programming Guideline, Stack Exchange

I. INTRODUCTION

Stack Overflow (SO) is a program-related Q&A website. Software developers use SO to initiate, browse, and answer questions. As of 2022, SO has 17.16 million registered users and over 50 million posts, 64% of the posts in SO contain code snippets [3]. However, these code snippets are not fully compliant with coding guidelines and are likely to have quality problems [9], since coding style compliance is highly related to the readability and maintainability of the source code [4]. Simply copying (or reusing) them may cause errors as well as software maintenance issues [10]. Currently, there is still a little research about coding style compliance on SO. Therefore, we carefully plan an empirical study to bridge this research gap.

We first collected 106,248 Python code snippets, 98,723 C/C++ code snippets, and 110,304 JavaScript code snippets on SO. Then, we explore the violation ratio and most frequently found violation type based on the collected dataset. The experimental results illustrate that 93.54% Python code snippets, 98.95% JavaScript code snippets, and 91.13% C/C++ code snippets in question-type posts contain coding style violations, while 89.53% Python code snippets, 100% JavaScript code snippets, and 65.83% C/C++ code snippets in answer-type posts contain coding style violations. The most frequently found coding style violation is non-standard space indentation for JavaScript and C/C++ and syntax error for Python. The Pearson correlation analysis indicates a moderate negative correlation (−0.8 ≤ r ≤ −0.6) between the number of code violations per statement and the “Score” of Python and C/C++ posts, the “FavoriteCount” of C/C++ questions, and the “CommentCount” of JavaScript questions.

The findings of our research suggest that the majority of code snippets on SO do not comply with proper coding style guidelines. As code violations can decrease readability and maintainability, developers should be cautious when reusing code snippets from SO. Especially, users should focus on syntax errors and inconsistent indentation styles, which constitute the majority of the total violations. In addition, when raising questions on SO, we suggest that users should pay more attention to their coding style compliance because a better coding style tends to receive more comments, and code snippets with fewer violations have a greater chance to get a higher “Score”.

II. RELATED WORK

Some previous studies have concentrated on coding style compliance on Stack Overflow (SO). The work of Hart et al. [14] sought to ascertain the influence of social reputation and other aspects on the perception of answer quality. Their findings suggested that social reputation had no substantial effect, while the presentation styles of completeness and conciseness were deemed to be more influential. The research presented herein elucidates the relationship between coding style compliance and the quality of posts, thereby demonstrating the potential for further explorations.

Treude et al. [17] conducted a study of the extent to which developers appraise SO code snippets as self-explanatory. Additionally, they probed the information absent from snippets and judged it not to be self-explanatory. The findings indicated that fewer than half of the code snippets in the sample were deemed self-explanatory. The primary coding style problems that impinge on the understandability of code snippets include incomplete snippets, code quality, missing rationale, code organization, clutter, naming issues, and missing domain information. The findings of their research demonstrate the

DOI reference number: 10.18293/SEKE2023-125
deleterious effect that coding style violations have on the quality of code.

More recently, Bafatakis et al. [18] investigated coding style compliance in Python code snippets on SO. Their research focuses on Python code snippets. Their results showed that 93.87% of snippets contain style violations, with an average of 0.7 violations per statement. Additionally, they found that user reputation seems to be unrelated to coding style compliance. While there is a strong correlation between the vote “Score” a post received and the average number of violations per statement. The authors also mentioned that the choice of languages and attributes could be expanded in further study.

Our work is an extension of Bafatakis et al.’s study. Based on their work, this study examines coding style compliance on SO as well, but with a particular focus on different types of posts (questions and answers). Furthermore, our research investigates the compliance of code snippets in more languages, as well as the types of violations in each language and the correlation between the violation rate and the attributes of “Score”, “ViewCount”, “FavoriteCount”, “AnswerCount”, and “CommentCount”.

III. STUDY DESIGN

In this section, we first present our research questions (RQs). Then, we select representative programming languages and the corresponding analysis tools. Next, we retrieve, sort, and filter Stack Overflow (SO) to construct our dataset. Finally, for each RQ, we provide a brief introduction of our methodology.

A. Research Questions

To determine the coding style compliance on SO, we will analyze JavaScript, Python, and C/C++ code snippets to answer the following RQs:

- RQ1. How many code snippets on SO contain coding style violations?
- RQ2. Which coding style rules are most frequently broken on SO?
- RQ3. Does coding style compliance correlate with SO attributes?

B. Programming Languages and Code Analysis Tools

In our research, Python, C/C++, and JavaScript were selected as the research objects to ensure a comprehensive study, since these languages are among the top ten most popular languages on SO [1], which can provide enough data to support in-depth research.

Corresponding to the three languages, we chose Pylint, Cpplint, and ESLint to check the coding style compliance. In the case of Python, we selected Pylint\(^1\) as it is a widely-used tool that follows the style recommended by PEP 8 (a Python style guide). Pylint provides a comprehensive set of checks that analyze code for potential errors and code smells. For C/C++, we chose Cpplint\(^2\), which is based on Google’s coding style guides and has a strict set of rules (i.e., naming conventions

\(^1\)https://pypi.python.org/pypi/cpplint
\(^2\)https://github.com/google/cpplint

and formatting rules) that help developers write consistent and readable code. Finally, for JavaScript, we selected ESLint\(^3\), which is a popular tool for identifying problematic patterns in JavaScript code. We chose these specific code analysis tools based on their ability to analyze code for potential issues and enforce coding standards. Each tool has its own set of configurable rules and guidelines that can be customized to meet the specific needs. Moreover, these tools are widely used in their respective communities and can provide reliable and effective code analysis.

C. Dataset Construction

The code snippets used in this study were acquired from Stack Exchange\(^4\), a network of various Q&A websites, with SO being one of the most actively visited ones.

We extracted question-type posts from July 2008 to December 2021, and their corresponding answers (from July 2008 to April 2022). The posts are distinguished by tags, e.g., posts about Python are tagged with “Python”. We retrieved Python, C/C++, and JavaScript code snippets and built our dataset by searching and downloading posts with certain tags.

In total, we extracted 106,248 Python code snippets, 98,723 C/C++ code snippets, and 110,304 JavaScript code snippets from question-type posts, and 40,418 Python code snippets, 44,316 C/C++ code snippets, and 47,463 JavaScript code snippets from answer-type posts.

Note that our research assumed that SO users have different programming experiences and ability levels; beginner programmers tend to ask questions (question-type posts) while experienced programmers prefer providing solutions (answer-type posts). Such differences may lead to discrepancies in the degree of coding style compliance. Consequently, we purposely divided the collected dataset into two subgroups, namely question-type posts and answer-type posts. Our subsequent analyses were conducted on these separate subgroups respectively.

D. Analysis Method

In order to answer RQ1, code analysis tools were utilized to identify coding style violations in the collected Python, C/C++, and JavaScript code snippets. The violation ratio was subsequently calculated, which pertains to the number of coding style violations per statement for each language.

To address RQ2, we first classified hundreds of coding style violations into several categories according to the websites of Pylint, Cpplint, and ESLint, as shown in Table II. Although these code analysis tools have different classification criteria for coding style violations, upon further analysis, we noticed a significant overlap in violation types across different programming languages (e.g., unused-import and reimported in Python, no-duplicate-imports and no-restricted-imports in JavaScript, build/include and build/include-order in C/C++). Based on this finding, we decided to consolidate and discuss some of the most prominent categories of violations, in order

\(^3\)https://eslint.org/
\(^4\)https://stackexchange.com/
to facilitate fair comparison of common violations across different languages. They are indentation-namespace/name, include/import, and whitespace/space. The distribution of these categories is provided in Table III.

The improper naming of variables or functions can be identified as a “Name” violation. This type of violation is crucial because clear names can help developers understand and work with the code better. Failure to follow standard naming conventions can cause confusion, make debugging harder and sometimes lead to program errors. Examples of these violations in Python include duplicate-argument-name (i.e., duplicate argument names in function definitions), arguments-renamed (i.e., a method parameter has a different name than in the implemented interface or an overridden method), bad-dunder-name (i.e., a dunder method is misspelled or defined with a name, not within the predefined list of dunder names), etc.

“Import” contains violations regarding importing. While many users might not immediately encounter issues due to inadequately importing, those who want to reuse code snippets would likely come up against difficult-to-diagnose bugs caused by such violations. Examples of this type of violations in Python include import-self (i.e., a module is importing itself) and shadowed-import (i.e., a module is aliased with a name that shadows another import). While in JavaScript, there are sort-imports (i.e., enforce sorted import declarations within modules) and no-duplicate-imports (i.e., disallow duplicate module imports).

“Space” contains violations regarding indentations. Indentations play a vital role in readable code, especially in languages like Python where correct indentation impacts code functionality. As a result, Python tends to have fewer indentation errors compared to languages such as C/C++. Common C/C++ indentation violations involve whitespace/braces, whitespace/comma, whitespace/empty-if-body, whitespace/semicolon, etc. Although these violations do not affect program operation and might seem insignificant, they create cluttered code, making it more challenging for colleagues to decipher intent. Thus, the importance of adhering to the standard of indentations should not be underestimated, as it can significantly improve the readability and maintainability of the code.

To answer RQ3, we retrieved five attributes of question-type posts, i.e., “Score” (the number of upvotes minus downvotes a post received), “ViewCount” (the number of times a post was viewed), “FavoriteCount” (the number of times a post was saved by other users), “AnswerCount” (the number of answers a post received), and “CommentCount” (the number of comments a post received). Besides, we retrieved two attributes of answer-type posts, i.e., “Score” and “CommentCount”. We first used scatter plots to visualize the relationship between the violation ratio and the attribute values. After that, we preprocessed the collected data and removed the outliers using the interquartile range (IQR) method. Our study partitioned the collected posts into multiple groups based on the variance in violation ratios with increments of 0.2. Subsequently, the mean attribute values of each group were computed. The Pearson correlation coefficient was employed to explore the relationship between the violation ratio and the mean attribute values of each group.

IV. EMPIRICAL RESULTS

We present our findings regarding each research question.

A. Results of RQ1

As shown in Table I, code snippets without any coding style violations are uncommonly found in JavaScript. For Python and C/C++ code snippets, it is noted that the prevalence of compliant code snippets in answer-type posts outweighs that in question-type posts. In the case of C/C++ code snippets, the ratio of compliant code snippets in answer-type posts exceeds that in question-type posts by 25.31%.

We also calculated the violation ratio. Our analysis revealed that the violation ratios are significantly elevated across all three languages, with a particularly pronounced increase in the case of JavaScript.

In conclusion, more than 80% of the code snippets on Stack Overflow (SO) contain coding style violations, indicating that code snippets on SO do not have good coding style compliance, and it is not recommended to reuse them without careful inspection.

B. Results of RQ2

Based on the classification method on the official websites of the code analysis tools, we calculated the proportion of each category of coding style violations in the three languages. As shown in Table II, the most frequently found violation is
TABLE I: Coding Style Violations in Question-type and Answer-type Posts

<table>
<thead>
<tr>
<th>Language</th>
<th>Code Snippets with Violations</th>
<th>Violation Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Question</td>
<td>Answer</td>
</tr>
<tr>
<td>Python</td>
<td>93.35%</td>
<td>89.53%</td>
</tr>
<tr>
<td>C/C++</td>
<td>91.14%</td>
<td>65.80%</td>
</tr>
<tr>
<td>JavaScript</td>
<td>98.95%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

syntax error for Python and whitespace/braces for C/C++. For JavaScript code snippets, the most frequently found violation is indent (i.e., enforce consistent indentation) for questions and semi (i.e., require or disallow semicolons instead of ASI) for answers.

To facilitate further comparison, we identified common violations in the three languages and subsequently categorized these violations into three distinct groups. It can be seen in Table III that a significant number of coding style violations in C/C++ and JavaScript are related to indentations, whereas violations in Python code snippets involve a variety of aspects.

The main reason for this difference is the nature and design of the programming languages. C/C++ and JavaScript are typically written with curly braces where code blocks are enclosed in them. Due to the visual clarity produced by these curly braces, proper indentation is not required for code to be syntactically correct. Nevertheless, consistent indentation remains crucial for code legibility and readability. Therefore, most coding style guidelines for C/C++ and JavaScript emphasize consistent indentation as the primary formatting guideline.

On the other hand, Python is designed to use indentation to define its code blocks. This means that indentation is not just a matter of style, but it is an essential part of the language syntax. In Python, improper indentation can result in syntax errors that make the code unexecutable. As a result, Python coding style guidelines cover multiple aspects of indentation and whitespace usage, such as the number of spaces per indentation level, the use of tabs versus spaces, and the use of whitespace in other contexts such as line breaks and wrapping.

Additionally, Python has a more extensive set of language constructs and syntax features compared to C/C++ and JavaScript. This means that Python coding style guidelines need to cover a wider range of aspects beyond simple indentation rules, such as variable naming conventions, function and class definitions, control structures, etc. The goal of these guidelines is to promote consistency and readability across Python code, especially in larger and more complex projects.

C. Results of RQ3

Figure 2 displays data on Python posts that are questions. Most of these posts received a low “Score” and a low violation ratio. Some posts got a high “Score” with a low violation ratio, while others got a low “Score” with a high violation ratio. Only a very small number of posts in this category got a high “Score” with a high violation ratio. Similar Zipfian distributions were observed for answer-type posts and posts in two other languages.

We posit that a correlation coefficient surpassing the threshold value of $|r| > 0.6, p < 0.01$ represents a significant correlation between the two variables under consideration. Accordingly, we found that whether a code snippet is compliant with coding style rules correlated with some SO attributes. For instance, Figure 3 illustrates that the “Score” for answer-type C/C++ posts exhibits a moderate correlation with the violation ratio. The mean values of “Score” have a Pearson correlation coefficient of $r = -0.789, p = 0.002$ with the violation ratio. The result means that for C/C++ questions on SO, the ones with code snippets following coding style guidelines tend to receive a higher “Score” than other questions. As shown in Table IV, similar results were observed across some of the other attributes.

For JavaScript posts, it can be observed that the correlation between attribute values and violation ratio is commonly found to be insignificant, except the “CommentCount” attribute in questions. The findings of RQ2 indicate that the majority of violations in JavaScript code do not significantly affect how the program operates. This may lead users to perceive coding style violations in JavaScript posts as less important.

It can be noted that violations have a greater impact on “Score” than other attributes. Users tend to upvote more on questions and answers with fewer coding style violations and believe them to be of higher quality. Besides, we noticed that the violation ratio of question-type C/C++ posts is negatively correlated with “FavoriteCount”.

At this time, there is no conclusive evidence proving that coding style violations significantly affect users’ browsing, commenting and answering behaviors. As all “ViewCount”, “AnswerCount” and answers’ “CommentCount” exhibit weak correlation ($|r| < 0.6$) with the violation ratio, as shown in Table IV, which means that under certain circumstances, users seem to pay less attention to the coding style compliance. It is possible that users are willing to overlook minor coding style violations if the code is otherwise useful or valuable to them. They may prioritize the benefits of the code over its compliance with coding style guidelines.
TABLE II: Proportion of Each Category of Coding Style Violations

<table>
<thead>
<tr>
<th>Python Violation Types</th>
<th>C/C++ Violation Types</th>
<th>JavaScript Violation Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Refactor for a “good practice” metric violation</td>
<td>1 build: builds when the error</td>
<td>1 Possible Errors: Rules related to possible syntax or logic errors in JavaScript code</td>
</tr>
<tr>
<td>2 Convention for coding standard violation</td>
<td>2 readability: readability error</td>
<td>2 Best Practices: Rules related to better ways of doing things to help you avoid problems</td>
</tr>
<tr>
<td>3 Warning for stylistic problems, or minor programming issues</td>
<td>3 runtime: runtime error</td>
<td>3 Parsing error: parsing error</td>
</tr>
<tr>
<td>4 Error for important programming issues</td>
<td>4 whitespace: space indentation error</td>
<td>4 Variables: Rules related to variable declarations</td>
</tr>
<tr>
<td>5 Fatal for errors which prevented further processing</td>
<td></td>
<td>5 Node.js and CommonJS: Rules related to code running in Node.js, or in browsers with CommonJS</td>
</tr>
<tr>
<td>6 Informational messages that Pylint emits</td>
<td></td>
<td>6 Stylistic Issues: Rules related to style guidelines, and are therefore quite subjective</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 ECMAScript 6: Rules related to ES6, also known as ES2015</td>
</tr>
</tbody>
</table>

TABLE III: Proportion of Each Reclassified Category of Coding Style Violations

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>C/C++</td>
</tr>
<tr>
<td>name</td>
<td>import</td>
</tr>
<tr>
<td>C/C++</td>
<td>0.05%</td>
</tr>
<tr>
<td>Python</td>
<td>14.01%</td>
</tr>
<tr>
<td>JavaScript</td>
<td>0.01%</td>
</tr>
</tbody>
</table>

TABLE IV: Pearson Correlation Coefficients between Different Attributes and the Violation Ratio

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Python</th>
<th>C/C++</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question</td>
<td>Score</td>
<td>-0.615</td>
<td>-0.789</td>
</tr>
<tr>
<td>ViewCount</td>
<td>0.305</td>
<td>0.517</td>
<td>0.395</td>
</tr>
<tr>
<td>FavoriteCount</td>
<td>0.348</td>
<td>-0.627</td>
<td>0.130</td>
</tr>
<tr>
<td>AnswerCount</td>
<td>0.186</td>
<td>0.127</td>
<td>0.483</td>
</tr>
<tr>
<td>CommentCount</td>
<td>0.086</td>
<td>0.529</td>
<td>-0.649</td>
</tr>
<tr>
<td>Answer</td>
<td>Score</td>
<td>-0.607</td>
<td>-0.745</td>
</tr>
<tr>
<td>CommentCount</td>
<td>0.487</td>
<td>0.521</td>
<td>0.347</td>
</tr>
</tbody>
</table>

Fig. 3: Mean value of “CommentCount” for question-type C/C++ posts with the same range of violation ratio

V. DISCUSSION

A. Implications

Our findings have important implications for both developers and software organizations.

For developers, our study highlights the importance of compliance with established coding styles and conventions in order to produce readable and maintainable code. The results show that code that does not conform to a standard style is more likely to receive negative feedback and be downvoted on SO, which can impact the reputation of the developer and reduce the visibility of their code. Therefore, developers should carefully follow established coding styles in order to improve the quality and maintainability of their code, as well as their position within the developer community. Automated tools such as linters and code formatters can help developers capture coding style violations in the early stage of the development process. Besides, developers should collaborate with their peers to improve their coding style. Code reviews can provide feedback on how to improve code readability and maintainability.

For software organizations, our study highlights the importance of establishing and enforcing coding standards across development teams. By ensuring that all developers adhere to a consistent coding style, organizations can improve the quality and readability of their code, as well as increase the efficiency and effectiveness of their development processes. Additionally, our study suggests that organizations should introduce tools and resources to help developers adhere to coding standards, such as automated code review tools and training programs.

Finally, our study has implications for the software development community. The findings suggest that the use of standardized coding styles and conventions can improve the quality and maintainability of code, which can ultimately lead to better software products and a more robust software industry.

B. Threats to Validity

Many code snippets used in our experiments are incomplete, which is a possible threat to the conclusion of our study. As many questions on SO only contain partial code, several coding style rules cannot be applied. To address this issue, we had to disable these rules, as illustrated in Table V. While this approach may have some impact on the results obtained, it is unlikely to introduce a significant bias in the study’s findings. Nonetheless, caution is warranted when interpreting the results, and future research could benefit from using more complete code snippets.
In the process of gathering data, code snippets were selectively extracted through the utilization of pertinent tags. Nonetheless, it is plausible that a subset of posts might contain erroneously labeled snippets, or code snippets written in Python, C/C++, and JavaScript are not labeled by those tags. As a mitigation strategy, we processed all code snippets with code analysis tools and found a small minority, less than 10%, of snippets that proved to be beyond the bounds of the tools. These snippets were deemed invalid and were consequently excluded by way of manual intervention. For instance, within the 116,000 snippets that were extracted with the “Python” tag, 9,752 snippets (8.41%) were identified as unsuitable for analysis and were consequently eliminated. Nonetheless, it is conceivable that some mislabeled code snippets may still exist.

VI. CONCLUSIONS

To explore the coding style compliance on Stack Overflow, we first collected over 400,000 Python, C/C++, and JavaScript code snippets from July 2008 to April 2022. Then, we conducted empirical experiments on the collected dataset. Our findings indicate that: 1) 93.54%, 91.13%, and 98.95% Python, C/C++, and JavaScript code snippets in questions and 89.53%, 65.83% and 100% Python, C/C++, and JavaScript code snippets in answers contain violations of coding style guidelines. 2) The most frequently broken rule is whitespaces in C/C++, undefined syntax errors in Python, inconsistent indentation and semicolon usage in JavaScript questions and answers. 3) Posts in Python and C/C++ with more violations have lower “Score”. For C/C++ questions, “FavoriteCount” is negatively correlated with violation per statement, while for JavaScript questions, “CommentCount” is negatively correlated with violation per statement.

In this study, we analyzed code snippets in Python, C/C++, and JavaScript on Stack Overflow to explore the problem of coding style compliance. In the future, we will further investigate other programming languages and Q&A websites to provide additional insights.

ACKNOWLEDGMENTS

This work was supported by the Spark Project of the Beijing University of Technology (Project No. XH-2023-02-35).

REFERENCES


Mining the Relationship between Object-Relational Mapping Performance Anti-patterns and Code Clones

Zeshan Xu†‡, Jie Zhu†‡, Li Yang∗†, Chun Zuo§
†Institute of Software, Chinese Academy of Sciences, Beijing, China
‡University of Chinese Academy of Sciences, Beijing, China
§Sinosoft Co., Ltd., Beijing, China
{xuzeshan21, zhujie212}@mails.ucas.ac.cn
yangli2017@iscas.ac.cn
zuochun@sinosoft.com.cn

Abstract—The use of Object-Relational Mapping (ORM) in software development has become increasingly popular due to its superiority to simplify database interactions. Despite the prosperous development of ORM code smells detection tools for general code smell problems related to coupling and cohesion, these tools do not capture issues that are specific to ORM code statement context. In this work, we fill the gap with the potential performance anti-patterns of repetitive ORM code by heuristic analysis and code clone analysis on 6 open source ORM systems in Java and Python (Saler, Wagtail, Zulip, Taiga, Protal, and Roller). For each occurrence of this code smell, we distinguished problematic instances that potentially require further fixes among justifiable ones. Through our research, we identified four anti-patterns associated with repetitive ORM code and proposed fix strategy for each of these anti-patterns. Additionally, our study delves into the relationship between repetitive ORM code anti-patterns and code clone, which reveals that a substantial proportion of repetitive ORM statements can be found in cloned code. Experiments show that repetitive ORM code can lead to a waste of system performance. This research highlights the impact of ORM code context on the proper use of ORM frameworks and emphasizes that copying ORM code without context evaluation can be detrimental to system performance.

Index Terms—code clone, anti-pattern, code smell, object-relational mapping, static analysis, empirical study

I. INTRODUCTION

As a well-established programming technique, Object-Relational Mapping (ORM) has emerged as a solution to the problem of Object-Relational Impedance Mismatch [1], practitioners employ ORM frameworks to close the divide between databases and applications by handling the tasks of data mapping and persistence [2], [3]. However, traditional static code analyzers are insufficient in capturing the unique attributes of ORM code and thus unable to identify potential issues within ORM code [4]. Rahman et al. reported that 76 of 77 projects don’t follow the rule recommendations of Hibernate architecture [5]. Rahman et al’s latest study identified and defined a greater number of code smells within the field of ORM [6]. However, previous approaches are still coarse-grained and only consider the appropriateness of ORM statement only, while operations involving ORM persistent objects, such as querying, saving, modifying, and deleting, all entail contextual information [3]. In other words, we consider that utilizing ORM statements requires taking into account the contextual information of code. Although each ORM statement may be innocuous on its own, duplicating it without considering the context is likely to result in unnecessary performance overhead of the new context. Fig 1 shows an example of repetitive ORM code from Saler. The developer defined the exact same ORM statement in both interfaces, but used different interface names. findByUsername sounds from the name to find a single piece of data through Username, while findAllByUsername finds all data through Username. When the developer only wants to obtain a piece of data, using the findByUsername interface will lead to a lot of unnecessary query data, resulting in a waste of ORM system performance.

Fig. 1. A repetitive ORM code example. The ORM framework part of the code is marked in blue, and the differences are marked in red.

To help developers improve ORM practices, in this paper, we focus on studying repetitive ORM statements. We conducted heuristic analysis on 6 open source ORM systems written in Java and Python. Our investigation revealed four repetitive ORM code anti-patterns.

Intuitively, repetitive ORM statements could be related to,
or are even a consequence of code clones. During software development, developers are often tasked with performing unfamiliar programming tasks [7]. When confronted with these challenges, developers frequently resort to online searches or scouring through project code. In the search results, code snippets are the ideal resources for developers to leverage during the development process [8], especially for unfamiliar tasks. A code snippet refers to a segment of code that achieves one or multiple specific programming objectives [9]. Code snippets can be directly reused by copy-pasting [10], which may contain repetitive ORM code statements.

Fig 2 presents the overall framework of our research. We examined 177 open source ORM systems written in Java and Python, using a Criticality Score [10] metric to guide our choices, and performed detailed heuristic inspections on six of them (Saler, Wagtail, Zulip, Taiga, Protal, and Roller) to identify recurring ORM anti-patterns. To further investigate the relationship between repetitive ORM fragments and code clone, we used NiCad [5], a code clone detection tool, to analyze each revision of these 6 projects. Combined with our heuristic analysis, we found that code clone resulted in a large number of repetitive ORM fragments. These results suggest that code clone may be a significant contributing factor to the presence of repetitive ORM fragments in software systems.

Specifically, we make the following contributions:

• We uncovered four new ORM performance anti-patterns through a comprehensive heuristic analysis of over 1K instances of repeated ORM code statements in six open-source ORM systems with suggested fixes for each anti-pattern.

• We found that code clone can lead to performance anti-patterns in ORM systems, and the majority of problematic repetitive ORM code anti-patterns (90.2%) were found to exist in cloned code fragments.

• We discovered that 73% of the instances of the repetitive ORM code anti-patterns that were not detected by code clone detection tools were actually from microcloned fragments. This finding highlights the need for future studies to investigate the negative impact of microclone on system performance.

• We found that reckless code clone can lead to performance anti-patterns for ORM frameworks and discovered 153 ORM performance anti-patterns in six large open source systems.

All data are publicly available\(^1\).

II. RELATED WORK

A. Empirical Studies on ORM System

Several studies have investigated Object-Relational Mapping (ORM) systems [4], [11], with Chen et al. [4] examining open source ORM systems and highlighting the hidden maintenance costs of using ORM frameworks. Meanwhile, SAddAR et al. [11] conducted empirical research on the performance of ORM frameworks. These studies have shown that although there are several benefits of using an ORM framework, maintaining ORM code can pose certain challenges. Specifically, Chen et al. [4] found that ORM code undergoes more frequent changes than regular code and lacks automated verification and detection methods, based on their analysis of different revisions of open source systems. In contrast, our study focuses on repeated ORM statements and code clone in the system, incorporating context (i.e., surrounding code) to find and identify methods for addressing ORM code issues.

B. Code Smells and Anti-patterns

Code smells and anti-patterns can be indicators of poor design and implementation of code, which can adversely affect the maintainability [11]–[14], understandability [15], [16], and performance [17] of a software system. To address their effects, several studies have proposed detecting code smells and anti-patterns [18]–[20]. In the context of the ORM system that we investigated, Holder et al. [21] proposed a metric suite to measure the complexity of ORM mapping code, while Silva et al. [5] suggested a set of rules to verify whether Hibernate entity code follows the JPA [22] specification. Loli et al. [23] surveyed previous research [24]–[26] and proposed ORM code smells in the literature, which surveyed developers’ agreement on the definition of smell. The findings indicated that most developers agreed with the definition and severity of the smells.

Code clone, or repetitive code, is a code smell that can arise when a developer copies and pastes a piece of code from one place to another [27]. This code clone method may lead to software quality problems [28]–[30], and there are some works in previous research committed to detect and solve these odors through various methods [5], [6], [16], [31].

III. METHODOLOGY

In this section, we will describe our methodology, which includes two parts: (1) How we identify repetitive ORM

\(^1\)https://figshare.com/s/e5a20d2267b08a39018e
statements in existing open source ORM systems for heuristic research and (2) How we conduct heuristic research to investigate which code repetitive ORM statements have anti-patterns.

A. Studied Systems

We manually analyzed six of open source ORM system in table I. To generate a practical dataset, We combined the dataset of 77 projects mentioned in previous research on ORM systems [5], [6] with 100 open source projects using ORM on github. we used the open source project importance score2 to screen and evaluate projects. To eliminate the relationship between the ORM framework and the programming language, we collected not only Java projects but also Python projects, primarily those using the Django and Hibernate framework.

B. Define Repetitive ORM Statements

In this paper, we define repeated ORM statements as statements with the same conditional call structure within the ORM framework. For instance, we consider the two ORM statements below to be duplicates:

\[
\text{Product.objects.filter(vector = None).}
\]
\[
\text{prefetch_related(“data”).order_by()}
\]
\[
\text{Product.objects.filter(document = “”).}
\]
\[
\text{prefetch_related(“addresses”).order_by()}
\]

C. Identify Repetitive ORM Statements

We employed static analysis to analyze the source code. We defined ORM statements as an abstract combination of models, methods, and parameters, which can exhibit various forms of queries, additions, deletions, and modifications based on their intended purposes. We excluded repetitive ORM statements used for model building and extracted parameter information, such as table and column names, to facilitate heuristic analysis.

IV. ANTI-PATTERNS OF REPEATED ORM CODE

Similar to previous research on anti-patterns, we consider repetitive ORM anti-patterns as "superficial indicators of deeper problems in the system" [32]. The presence of repeated ORM code may indicate an underlying problem that requires attention. However, not all instances of repetitive ORM code are problematic. We categorized each repetitive ORM statement as either problematic or reasonable to ignore based on the context of the repetitive ORM code (i.e., the surrounding code). Our research can help developers improve their ORM coding practices and inspire future research in this area.

In total, we examined 393 groups of repetitive ORM statement pairs, constituting more than 1K ORM statements in aggregate. Each group comprised two or more ORM statements with matching conditional call structures. In total, we discovered 153 problematic instances in the ORM code, featuring four recurring anti-patterns: 1) Selecting superfluous data (Select All), 2) Retrieving related data that remains unused (Associated Object), 3) Employing redundant sorting procedures (Order Waste), and 4) Repeatedly fetching non-updated data (Cache Waste). It is noteworthy that a single set of repetitive ORM code pairs could encompass multiple issues. Specifically, we observed these recurring anti-patterns occurring 138, 26, 14, and 10 times, respectively.

Table II displays the number of instances of each anti-pattern that we identified manually. We will provide a comprehensive discussion of each anti-pattern and propose suitable solutions below.

Select all. In our investigation, we discovered that when a repeated ORM query statement selects entity objects from the DBMS, the original query selects all columns of an object, while the cloned repeated ORM statement only uses a small number of columns. For instance, let us consider the Proxy class. If the code cannot use all the columns of the Proxy, it is recommended to add filter column conditions before the query statement. As the ORM framework lacks knowledge about the required data, it selects all columns by default, leading to avoidable performance degradation. Fix Strategy: Customize the specific parameters of the ORM query based on the context, and include the appropriate column tag in the statement.

\[
A = \text{ProxyModel.objects.filter(}
\]
\[
\text{cluster_name = “A”, dc = “lf”, service = “nsq”)}
\]
\[
ip, port = A.ip, A.port
\]
\[
A = \text{ProxyModel.objects.filter(}
\]
\[
\text{cluster_name = “A”, dc = “lf”, service = “nsq”)}
\]
\[
ip = A.ip
\]

Associated object. This anti-pattern indicates that associated object data has been queried unnecessarily. The problem with this anti-pattern is that it retrieves unnecessary tuples from other database tables. When using the ORM framework, developers can specify the relationship between entity classes, such as one-to-one, one-to-many, many-to-one, and many-to-many. The ORM framework provides different optimization techniques to specify how to obtain associated entity objects from the database. For example, in the code, if only the Users information is needed, but the ORM framework retrieves the associated Books data, then it causes unnecessary overhead in terms of performance. Retrieving users together with associated objects can be expensive, particularly when there are many Books in Users. Previous studies have shown that ORM frameworks often use SQL connections to obtain too much data, which can significantly reduce system performance [33]. Different ORM frameworks provide different methods to solve this redundant data problem, and our approach can provide developers with guidance to address such issues. Fix Strategy: Instead of directly utilizing the duplicated old model, establish a new model to access the necessary table.

Sort waste. This anti-pattern involves performing unnecessary sorting operations. Sorting the queried data is a commonly used technique by developers in ORM. There are two ways to

2https://github.com/ossf/criticality_score
sort data in ORM: using the order_by() method or specifying the sort order when defining the model. The default is ascending order, but it can be changed to descending order using the desc() method. And the column name for sorting could also be customized. Studies have shown that sorting operations can significantly degrade SQL performance [41], particularly when sorting tables that do not contain indexes. To further investigate this issue, we analyzed the time consumption of ORM statements with and without sorting operations using the database Explain function (i.e., the execution plan). Our results showed that redundant sorting operations accounted for 47% of performance waste. This finding highlights the importance of optimizing sorting operations in ORM coding practices. **Fix Strategy:** Evaluate the need for sorting based on the context, and remove unnecessary sorting actions.

**Cache waste.** Our investigation found that ORM frameworks offer support for caching [34], which involves sharing objects between transactions to improve performance. However, striking a balance between performance and data staleness can be challenging in distributed systems that use caching. For instance, consider the following example:

```python
RegionModel.objects.filter(id = 481).
update(“region” = “cn”)
...
RegionModel.objects.filter(id = 481).
values_list(“region”)
...
RegionModel.objects.filter(id = 481).
values_list(“region”)
```

The first query statement is necessary because the user data has been updated by another query for the same primary key in the filter clause. In contrast, the second query does not need a second query since the data has not changed. While most ORM frameworks provide caching mechanisms to reuse fetched data and minimize database access, the caching configuration is not automatically optimized for different application systems [34], and some ORM frameworks even disable caching by default. Even if the retrieved entity object has not been modified, this repeated ORM statement may execute millions of times in a short period. Although different ORM frameworks may have distinct configurations, the issues we report are common, and our finding can aid developers in optimizing caching. **Fix Strategy:** Remove the redundant ORM code that does not modify the data, and utilize the same object generated by the context for accessing purposes.

### Table I. Projects in our research

<table>
<thead>
<tr>
<th>System</th>
<th>About</th>
<th>Total lines of code</th>
<th>Language</th>
<th>Stars</th>
<th>Commits</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saler</td>
<td>Saleor Core: The high performance, composable, headless commerce API</td>
<td>435882</td>
<td>Python</td>
<td>18.2k</td>
<td>20284</td>
<td>3.12.3</td>
</tr>
<tr>
<td>Wagtail</td>
<td>A Django content management system focused on flexibility and user experience</td>
<td>198763</td>
<td>Python</td>
<td>13.4k</td>
<td>14805</td>
<td>4.2.1</td>
</tr>
<tr>
<td>Zulip</td>
<td>Open-source team chat that helps teams stay productive and focused</td>
<td>258309</td>
<td>Python</td>
<td>17.4k</td>
<td>50313</td>
<td>6.1.0</td>
</tr>
<tr>
<td>Taiga</td>
<td>Agile project management platform. Built on top of Django and AngularJS</td>
<td>113156</td>
<td>Python</td>
<td>5.8k</td>
<td>4164</td>
<td>1.0.0</td>
</tr>
<tr>
<td>Protal</td>
<td>Devproof Portal - Available modules Blog, Articles, Downloads, Bookmarks</td>
<td>53725</td>
<td>Java</td>
<td>14</td>
<td>971</td>
<td>1.0.0</td>
</tr>
<tr>
<td>Roller</td>
<td>Java-based open-source blog server that uses Hibernate for database interactions</td>
<td>96954</td>
<td>Java</td>
<td>116</td>
<td>4722</td>
<td>1.0.0</td>
</tr>
</tbody>
</table>

In our investigation, we identified 153 instances of ORM code anti-patterns in 1301 cloned blocks. Our findings suggest that reckless code clones can result in performance anti-patterns for ORM frameworks. This highlights the importance of identifying and addressing such anti-patterns to ensure optimal performance of ORM systems.

### Table II. Number of problematic instances found by our research

<table>
<thead>
<tr>
<th>System</th>
<th>Select All</th>
<th>Associated Object</th>
<th>Order Waste</th>
<th>Cache Waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saler</td>
<td>62</td>
<td>6</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Wagtail</td>
<td>13</td>
<td>8</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Zulip</td>
<td>32</td>
<td>3</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Taiga</td>
<td>15</td>
<td>5</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Protal</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Roller</td>
<td>12</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>138</td>
<td>26</td>
<td>14</td>
<td>10</td>
</tr>
</tbody>
</table>

### V. Relation between repetitive ORM code anti-patterns and code clone

#### A. Motivation

Code clone, or duplicating code, is a development pattern that is generally considered harmful to software maintainability, understandability, and performance. Previous research has focused on studying code clones in source code and understanding their impact. However, since cloning is often done in a hurry and without paying much attention to code context, ORM-related code may be copied as well. In the previous section, we identified four performance anti-patterns (i.e., Select All, Associated Object, Order Waste and Cache Waste) that can result from repetitive ORM code. However, not all repetitive ORM code is necessarily the result of code clone. In this section, we investigate the Relationship between Object-Relational mapping Performance anti-patterns and code clones. By doing so, we hope to provide developers with better practices for ORM code and encourage further research on code clone in this context.
B. Method

We use NiCad to detect clones, which has high precision (95%) and recall (96% identical). NiCad detects all major types of clones, including exact (type 1) and near-miss (types 2 and 3) clones, and is actively maintained (with the latest version released in November 2020). Clone detection results may vary based on different detector settings, so choosing appropriate parameters is important. In our research, we set the granularity of source code units to block level and used NiCad to detect block clones with a minimum size of 10 LOC and a similarity threshold of 70%, as recommended in a previous study [35]. This approach provided better clone detection results in terms of precision and recall.

We utilized NiCad to perform experiments on the open-source systems of the six studies mentioned earlier. Subsequently, we analyzed the clone detection results and compare the locations of clones with those of problematic instances. If two or more cloned snippets contained the same set of instances, we regarded those instances as related to the clone. Through manual screening and analysis, we classified the results of code clone into two categories: clone blocks that involved database calls and those that did not. To mitigate the impact of false negatives, we conducted additional heuristic research on all instances that NiCad failed to identify as clones. Our focus was on blocks of code that surrounded ORM statements and exceeded a threshold of 10 lines, which was the same as that of the clone detection tool.

C. Result

We found that code clone can lead to performance anti-patterns in ORM systems, and the majority of problematic repetitive ORM code anti-patterns (90.2%) were found to exist in cloned code fragments. Results revealed that 10% (15/153) of problematic occurrences of repetitive ORM code anti-patterns were identified as non-clones by automated code clone detection tools. To delve deeper into this issue, we scrutinized each undetected instance and found that 73% (11/15) were, in fact, derived from micro-cloned fragments. This observation underscores the need for future research to explore the detrimental effects of microclone on system performance and to develop guidelines for writing ORM code correctly. Our findings justify the need for future research to investigate the negative impact of microclone on system performance. In addition to considering code maintenance and refactoring overhead, future studies on code clone detection should also consider other possible side effects of code clone. Researchers can also optimize code clone detection methods by refining the detection of various types of statements commonly found in ORM code to improve its accuracy.

We found that 73% of instances of repetitive ORM code anti-patterns that were not detected by code clone detection tools were actually from microcloned fragments.

VI. Conclusion

ORM are widely used to solve the object-relational impedance mismatch problem by providing an object-oriented interface on top of a relational database, which enables easy saving and retrieval of program objects from secondary storage without mapping application data to database records. However, the quality of ORM code is often criticized by developers, especially its data persistence and query code. This paper presents an empirical study that investigates anti-patterns caused by duplication of ORM code (or code clone) in ORM systems. The study analyzes six open-source ORM systems (Saler, Wagtail, Zulip, Taiga, Portal, and Roller), identifies four new ORM code anti-patterns and proposed fix strategy for each of these anti-patterns, showing that repetitive ORM code can lead to a waste of system performance. These findings could serve as a valuable reference and provide direction for future research on ORM systems and optimization of ORM development practice standards.

The study also investigates the relationship between the repetitive ORM code anti-pattern and code clone. The results indicate that most problematic instances of repetitive ORM code occur in code clones, which are more likely to be microclones that are difficult to detect using existing code clone detection tools. This research highlights the impact of ORM code context on the proper use of ORM frameworks, emphasizing that copying ORM code without context evaluation can be detrimental. Therefore, future research should...
consider code context when providing guidance for ORM practices, and richer contextual information is required to construct new excellent ORM research data.

VII. Acknowledgements

We sincerely appreciate the valuable feedback from the anonymous reviewers. This work was supported by the Chinese Science and Technology Aid Project to Developing Countries (KY201906007) and the National Key R&D Program of China (No.2021YFC3340204).

REFERENCES


Contribution-based Test Case Reduction Strategy for Mutation-based Fault Localization

Haifeng Wang†, Kun Yang†‡, Xiangnan Zhao†, Yuchen Cui†, Weiwei Wang‡
†Center for Advanced Metering Infrastructure, National Institute of Metrology, China
‡College of Information Science and Technology, Beijing University of Chemical Technology, China
Email: wanghaifeng@nim.ac.cn

Abstract—Fault localization is one of the most expensive steps during software debugging. Mutation-based Fault Localization (MBFL) is a promising technique but with a high computational cost since the large mutation execution on mutation analysis. Previous studies mainly reduce the cost of decreasing the number of mutants and optimizing the process execution, such kinds of strategies have shown promising results. However, reducing the cost of MBFL by decreasing the number of test cases is also effective. In this paper, we propose a Contribution-Based Test Case Reduction (CBTCR) strategy for improving the efficiency of MBFL. CBTCR first measures the contribution value of each test case and then selects the test cases according to the value. Then it takes the reduced test suite for executing the mutants. We evaluate CBTCR on 383 real software faults from the Defects4J benchmark. The results show that CBTCR outperforms the other MBFL test case reduction strategies (e.g., FTMES, IETCR) in terms of the Top-N and MAP metrics. Also, CBTCR can significantly reduce 85.43% of the cost on average while maintaining a similar accuracy to original MBFL techniques.

Keywords—software debugging; fault localization; mutation-based fault localization; test case reduction

I. INTRODUCTION

Software debugging is an expensive and difficult process that costs developers a large amount of time and effort [1]. The first step in debugging, called fault localization (FL), is to identify the root cause of observed failures. FL is a complicated and time-consuming task, as it is important in software maintenance and evolution. To alleviate the human effort in localizing a fault, various automated FL techniques have been proposed, e.g., spectrum-based [2] and mutation-based [3], [4] techniques.

Spectrum-based fault localization (SBFL), is one of the most widely studied techniques. SBFL considers the binary coverage of the program elements but with inherently limited fault localization accuracy. Recent studies have shown that mutation-based fault localization (MBFL) techniques can help improve the performance of fault localization and achieve a higher fault localization accuracy than the state-of-the-art SBFL techniques [5].

MBFL is based on mutation analysis and works by making syntactic changes to the program. Although the studies have shown that MBFL aims the localize accuracy improvement of suspicious elements list for fault software debugging, the computational cost is high for such techniques. Hence, researchers investigate optimization strategies for MBFL techniques, which can be divided into three groups: (1) reduce the mutants [1], [3], [5]; (2) reduce the test cases [6], [7]; and (3) optimize the execution [8]. However, other opportunities exist for optimization.

In this study, a novel test case reduction strategy named CBTCR (Contribution-Based Test Case Reduction) is proposed aiming at optimizing the mutation execution. CBTCR keeps both failed and passed test cases like IETCR [7] but is different from FTMES [6], which only employs the failed ones. In detail, CBTCR uses test cover and suspiciousness of SBFL to measure the contribution value of test cases for MBFL. Next, the test cases are executed with higher values and avoid the execution of the rest ones.

The contributions of this paper are as follows:

- This paper proposes a contribution-based test case reduction (CBTCR) strategy to reduce the mutation execution cost of MBFL.
- This paper reports an empirical study on 383 real-fault programs from Defects4J, and the results show that MBFL execution cost could be remarkably reduced while adopting CBTCR. At the same time, the fault localization accuracy is kept almost the same as the original MBFL.
- The scripts and dataset used in this paper are all available in the GitHub repository [1] to facilitate the replication of our study and evaluation of future work.

The rest of this paper is organized as follows. Section II summarizes the background and related work. Section III describes the details of our approach. Section IV illustrates the experimental setup and analyzes our experimental results. Finally, Section V summarizes our study with potential future work.

II. BACKGROUND AND RELATED WORK
A. Mutation-Based Fault Localization

Mutation-based fault localization is a well-studied technique that is based on mutation analysis. In mutation analysis, mutants are used to evaluate the quality of test cases based on their ability to distinguish the mutants’ behavior from that of the original program [3]. Mutants can be evaluated by suspiciousness using the MBFL formulas. MBFL works

DOI reference number: 10.18293/SEKE2023-180

1https://github.com/hfwiwiz/CBTCR
based on the assumption that mutants killed mostly by failed test cases have a connection with program faults. Recent studies [3] also demonstrated that MBFL could significantly outperform other types of fault localization techniques.

B. MBFL Reduction Strategies

MBFL achieves high accuracy of fault localization but faces huge computational costs on execution. Various strategies have been presented for reducing the cost of MBFL, which can be divided into three categories: (1) reduce the mutants [1], [3]–[5], [9]. SELECTIVE [5] selects the "sufficient" mutation operators for generating mutants. SMBFL [1] reduces the mutants by examining only the statements in the dynamic program slice. In another aspect, SAMPLING [3], SOME [9], and WSOME [4] are reducing the cost by decreasing the mutants from the mutant set. (2) reduce the test cases [6]. FTMES [6] uses only the set of failed test cases to execute mutants while avoiding the execution of passed test cases. IETCR [7] measures the information of test cases using entropy and selects a proportion of them. (3) optimize the execution [9]. DMES [8] contains two kinds of optimizations, i.e., mutation execution optimization and test case execution optimization.

III. Our Method

A. Contribution Value of Tests

In software engineering, a good quality software it is essential to rely on good test cases, which are more powerful to discover bugs in the programs. It is the classic objective of testing and a failed test is more powerful than a passed test in that it can help find defects. In mutation-based fault localization, the test suite has been run first for obtaining elements covered by fail tests, so it is easy to distinguish the passed and failed test cases before executing mutants. However, how to distinguish the good and bad of the passed test cases is unclear.

To fill this gap, in this paper, we define Contribution value to measure the possible fault localization effectiveness for MBFL. Here, we will clarify the meaning of basic terms and notations used in this paper, including programs, tests, coverage, and suspiciousness.

Definition 1: Test cover. Given a program \( P = \{s_1, s_2, ..., s_n\} \) with \( n \) elements, a statement \( s \in P \), a test

\[ t \in T = \{t_1, t_2, ..., t_m\} \]

\( \text{if } t \text{ covers } s, \text{ which is formalized by:} \)

\[ \text{Cover}(t, s) = \left\{ \begin{array}{ll} 1 & \text{true, if } t \text{ covers } s \\ 0 & \text{false, otherwise} \end{array} \right. \]

By definition, a test cover concisely represents whether the statement \( s \) has executed the test \( t \), which is the original of the spectrum-based fault localization.

For a statement \( s \), the suspiciousness of \( s \) using SBFL formulas is denoted as \( \text{Sus}(s) \) (the SBFL formula we called the contribution formula, Dstar [2] is used in this paper). Then we use Contribution value to quantitatively measure the contribution of a test \( t \) for locating the fault in program \( P \). The Contribution value, simply the \( cValue \), can be formalized as follows:

\[ cValue(t, P) = \sum_{i=1}^{n} \text{Cover}(t, s_i) \times \text{Sus}(s_i) \]

for a test \( t \in T \) and statements \( s_1, s_2, ..., s_n \in P \).

In other words, the coverage of tests and the suspiciousness of statements imply the quantitative quality of the test \( t \) for MBFL. If a test \( t \) has a larger \( cValue \) indicates that \( t \) can cover more statements with greater suspiciousness, which suggests that \( t \) is more valuable for MBFL.

B. The Workflow of CBTCR

Fig. 1 shows the workflow of our proposed CBTCR. At first, CBTCR executes the tests against the program to obtain test cover and results. Then, the tests are divided into passed and failed sets. In addition, the statements covered by failed test cases are obtained for generating mutants by seeding mutation operators. Next, CBTCR works by calculating the contribution value of passed test cases. After sorting and selecting test cases with higher contribution values, a reduced test suite is generated by combining passed and failed test cases. Finally, the reduced test suite is executed on the mutants to get killing information. The mutants’ suspiciousness are calculated and the maximum one is assigned to the statement. The MBFL sorts all statements in descending order based on their suspiciousness and returns a ranking list of all program elements.
IV. EXPERIMENTAL ANALYSIS

To investigate the performance of CBTCR, we perform experiments to address the following three research questions:

- **RQ1**: How does CBTCR perform compared to different test case reduction strategies (i.e., FTMES, IETCR) and the original MBFL techniques on fault localization effectiveness?
- **RQ2**: How much execution cost does CBTCR need when comparing other test case reduction strategies and original MBFL techniques?
- **RQ3**: How does CBTCR perform compared to different MBFL techniques?

**RQ1** compares the effects of different test case reduction strategies for MBFL techniques and the performance of such techniques using CBTCR. **RQ2** is designed to evaluate the efficiency of MBFL adopting different reduction strategies.

A. Experimental setup

**TABLE II**

<table>
<thead>
<tr>
<th>SUBJECT PROGRAMS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project</strong></td>
</tr>
<tr>
<td>Lang</td>
</tr>
<tr>
<td>Chart</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Math</td>
</tr>
<tr>
<td>Closure</td>
</tr>
<tr>
<td>Cli</td>
</tr>
<tr>
<td>Codec</td>
</tr>
<tr>
<td>Compress</td>
</tr>
<tr>
<td><strong>Total</strong></td>
</tr>
</tbody>
</table>

1) **Subject Programs**: We conduct the experimental studies on the real-world benchmark of Defects4J [10]. Table II shows the statistics of subject programs. In total, we considered 383 faults out of 461 faults in Defects4J (v2.0.0). The remaining versions are excluded from the experiments since the omission faults or the faults cannot be detected by the test suites.

2) **Evaluation Metrics and Implementation of Experiment**: In the experiments, we utilize four metrics (i.e., EXAM [4], Top-N [11], MAP [11], and MTP [4]) to evaluate the effectiveness of our proposed CBTCR. A fault localization technique with a lower EXAM, higher Top-N, and higher MAP demonstrates a better technique. In addition, cost reduction techniques with lower MTP indicate better efficiency [9]. Note that we take the average rank of the statements that shared the same suspiciousness to break the tie [12].

3) **Implementation of Experiment**: We choose three MBFL test case reduction strategies (FTMES [6], IETCR [7] and the random strategy noted as “RAND”) as the baselines. We use 10%, 20%, and 30% to sample the pass test cases followed by the previous studies [3], [4], [7]. We implement Metallaxis [3] (noted as “MCBFL”) and MCBFL-hybrid-avg [12] (noted as “MCBFL”) as MBFL techniques. In addition, we generate mutants by using the mutation tool Major and consider the mutation operators provided by this tool.

B. Results Analysis

**TABLE III**

<table>
<thead>
<tr>
<th>Technique</th>
<th>Strategy</th>
<th>Sampling ratio = 10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBFL</td>
<td>RAND</td>
<td>42.52%</td>
<td>17.71%</td>
<td>33.80%</td>
</tr>
<tr>
<td></td>
<td>IETCR</td>
<td>62.16%</td>
<td>34.52%</td>
<td>19.54%</td>
</tr>
<tr>
<td></td>
<td>FTMES</td>
<td>0.87%</td>
<td>18.36%</td>
<td>24.92%</td>
</tr>
<tr>
<td>MCBFL</td>
<td>RAND</td>
<td>22.37%</td>
<td>12.51%</td>
<td>20.33%</td>
</tr>
<tr>
<td></td>
<td>IETCR</td>
<td>28.21%</td>
<td>21.92%</td>
<td>15.38%</td>
</tr>
<tr>
<td></td>
<td>FTMES</td>
<td>26.27%</td>
<td>32.48%</td>
<td>33.92%</td>
</tr>
<tr>
<td>Average</td>
<td>RAND</td>
<td>32.44%</td>
<td>15.11%</td>
<td>27.06%</td>
</tr>
<tr>
<td></td>
<td>IETCR</td>
<td>45.18%</td>
<td>28.22%</td>
<td>17.46%</td>
</tr>
<tr>
<td></td>
<td>FTMES</td>
<td>45.18%</td>
<td>28.22%</td>
<td>17.46%</td>
</tr>
</tbody>
</table>

**TABLE IV**

<table>
<thead>
<tr>
<th>Technique</th>
<th>Strategy</th>
<th>Sampling ratio = 10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBFL</td>
<td>RAND</td>
<td>0.0346</td>
<td>0.0037</td>
<td>0.0286</td>
</tr>
<tr>
<td></td>
<td>IETCR</td>
<td>0.0110</td>
<td>0.0040</td>
<td>0.0185</td>
</tr>
<tr>
<td></td>
<td>FTMES</td>
<td>0.0246</td>
<td>0.0018</td>
<td>0.0225</td>
</tr>
<tr>
<td>MCBFL</td>
<td>RAND</td>
<td>0.0444</td>
<td>0.0093</td>
<td>0.0110</td>
</tr>
<tr>
<td></td>
<td>IETCR</td>
<td>0.0464</td>
<td>0.0063</td>
<td>0.0192</td>
</tr>
<tr>
<td></td>
<td>FTMES</td>
<td>0.0068</td>
<td>0.0217</td>
<td>0.0032</td>
</tr>
<tr>
<td>Original</td>
<td></td>
<td>0.8371</td>
<td>0.8220</td>
<td>0.8518</td>
</tr>
</tbody>
</table>

1) **Answer for RQ1 (The effectiveness of CBTCR)**: Table I shows that, on MBFL, CBTCR locates 122 faults at Top-5 with a sampling ratio of 10%, which is more than RAND (103), IETCR (93), and FTMES (120). We can find that...
FTMES can locate more faults at Top-1 (43). One possible reason is that some passed tests cannot help in detecting faults. Also, CBTCR performs better when sampling 20% and 30% passed test cases at the metric of MAP. Besides, CBTCR performs similarly to the original MBFL (noted as "Original") when sampling 30% passed test cases. On MCBFL, CBTCR outperforms the other three strategies in all cases. Moreover, Table III shows the accuracy improvement for each strategy with different sampling ratios compared with CBTCR at MAP. The last row shows that CBTCR improves other strategies ranging from 13.57% to 45.18% in fault localization accuracy on average. Besides, CBTCR locates more faults at the metric of Top-3(115) when the sampling ratio equals 10% than Original. In addition, the wilcoxon signed-rank test (at a confidence level of 95% and p-values less than 0.05 means significant) in Table IV shows that EXAM of MBFL techniques with CBTCR has a significant difference with other three strategies and the original MBFL. Therefore, we regard 10% as the trade-off sampling ratio for the CBTCR strategy.

2) **Answer for RQ2 (The cost of CBTCR):** Table V further presents the detailed average reduction ratios MTP of each strategy. From Table V we can see that CBTCR reduces execution cost varied from 69.74% to 89.47% and it reduces 85.43% of the cost on average. Besides, CBTCR costs more 8.84% than FTMES, but it improves 13.57% of the fault localization accuracy for FTMES on average (see Table III). Therefore, CBTCR is a trade-off strategy that obtains better fault localization effectiveness by losing some efficiency of running this strategy. Hence, there still exists room for boosting the efficiency of CBTCR.

![Fig. 2. MTP of CBTCR and other strategies at sampling ratio 10%](image)

### Table V

<table>
<thead>
<tr>
<th>Project</th>
<th>RAND</th>
<th>IETCR</th>
<th>FTMES</th>
<th>CBTCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lang</td>
<td>89.23%</td>
<td>89.23%</td>
<td>99.00%</td>
<td>89.23%</td>
</tr>
<tr>
<td>Chart</td>
<td>89.47%</td>
<td>89.47%</td>
<td>99.41%</td>
<td>89.47%</td>
</tr>
<tr>
<td>Time</td>
<td>87.64%</td>
<td>89.81%</td>
<td>97.05%</td>
<td>87.64%</td>
</tr>
<tr>
<td>Math</td>
<td>86.09%</td>
<td>86.09%</td>
<td>95.13%</td>
<td>86.09%</td>
</tr>
<tr>
<td>Closure</td>
<td>69.74%</td>
<td>73.64%</td>
<td>74.67%</td>
<td>69.74%</td>
</tr>
<tr>
<td>Cti</td>
<td>88.93%</td>
<td>88.93%</td>
<td>98.68%</td>
<td>88.93%</td>
</tr>
<tr>
<td>Codec</td>
<td>87.02%</td>
<td>87.02%</td>
<td>96.17%</td>
<td>87.02%</td>
</tr>
<tr>
<td>Compress</td>
<td>85.35%</td>
<td>85.35%</td>
<td>94.09%</td>
<td>85.35%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>85.43%</strong></td>
<td><strong>86.19%</strong></td>
<td><strong>94.27%</strong></td>
<td><strong>85.43%</strong></td>
</tr>
</tbody>
</table>

V. **CONCLUSION AND FUTURE WORK**

This paper proposes a novel contribution-based test case reduction (CBTCR) for mutation-based fault localization (MBFL). CBTCR measures the contribution value of test cases using test coverage and SBFL suspiciousness, and then the test cases with higher values are used for running the mutants. CBTCR is a strategy that uses both failed test cases and some valuable passed test cases. This paper evaluates CBTCR by conducting an empirical experiment on 383 faults from Defects4J. The experimental results demonstrate that CBTCR outperforms the previous test case reduction strategies (IETCR and FTMES). Also, the results indicate CBTCR can highly improve the efficiency of MBFL by reducing 85.43% of the cost on average, while maintaining almost the same fault localization accuracy as original MBFL techniques. In the future, we plan to investigate the theory of CBTCR and extend our strategy to more large-scale programs.

**ACKNOWLEDGMENT**

The work is supported by the National Quality Infrastructure of China (Grant no. 2022YFF0608103).

**REFERENCES**

Understanding the Pre-Contract Process of Small Software Projects

Anelis Pereira-Vale  
Computer Science Department  
University of Chile, Chile  
apereira@dcc.uchile.cl

Daniel Perovich  
Computer Science Department  
University of Chile, Chile  
dperovic@dcc.uchile.cl

Sergio F. Ochoa  
Computer Science Department  
University of Chile, Chile  
sochoa@dcc.uchile.cl

Abstract— The phase of pre-contract of software projects is highly relevant for providers and acquirers. This time consuming phase should be done in a cost-effective way, since the possibility of finishing in a software contract is uncertain. The literature reports some macro-activities to guide this process. However, they are coarse grained, and therefore, difficult to follow in practice by practitioners. Moreover, it is not clear how the industry deals with the pre-contract process; particularly, when addressing the pre-selling of small software projects. This article presents an exploratory study that involves 14 Chilean software companies, in which we performed qualitative research to understand how they conduct the pre-contract process of small projects. The results were arranged in a model, named the Pre-Contract Process (PCP) model, which identifies the most prevalent actors, activities, outcomes, and a workflow according to the best practices used by these companies. These results extend those reported in the literature; particularly, the PCP model contributes with more in depth information about how to conduct the macro-activities in practices. Although this process is still preliminary, it can be used to inform the design of pre-contract activities in software companies and educational institutions. Future software engineers and practitioners can take advantage of it to address their own project pre-contract processes.

Keywords— pre-contract process model; qualitative research; pre-selling activities; small software projects; Chilean software industry.

I. INTRODUCTION

As in the rest of the world, most software projects performed in Chile are small. Many of them are conducted by micro and small software companies that do not have an explicit project pre-contract process (also known as per-selling); i.e., they use implicit procedures with a structure that depends on the person in charge of conducting them.

This lack of an explicit process jeopardizes the capability of the provider to measure, improve and repeat the process, and also to train the newcomers. The literature reports some general proposals as guidelines for such a process, however, there is no evidence on the use or usefulness of these proposals in small project prospects.

In order to understand the way in which the companies deal with the pre-selling process of small software projects, we conducted an exploratory study on 14 Chilean software companies, involving 20 practitioners experienced in this area.

The study identifies the most prevalent components in the processes according to the involved organizations; e.g., the major activities, the eventual workflow, the participating roles, and the key information to gather.

We used the guidelines of Grounded Theory [1, 2, 3, 4] to collect, analyze the individual data, and finally generate the Pre-Contract Process (PCP) model. Before starting the data collection, we defined the following working hypothesis to guide the study: Providers use an ad hoc and implicit process to perform the pre-contract stage.

The study results indicate that, at high level, the PCP model has several similarities to the one proposed by Savolainen et al. [5]. However, PCP study gains on understanding of the fine-grain activities, the conditions to make the transitions between activities, and the dynamic of the process that make it reusable, evolvable, and transferable to particular work contexts in academia and industry. Therefore, the PCP model represents an advance on the state-of-the-art. It can also help improve the state-of-the-practice in the academia, but mainly, in the micro and small software companies.

Next section analyzes the main models reported in the literature to address the pre-contract process of software projects. Section III describes the exploratory study conducted to understand the process in the observed companies. Section IV presents the results of the study, which were arranged in the proposed PCP model. Section V analyzes the variability level of PCP, considering its nature of unstructured process. Section VI discusses the threats to the results validity, and Section VII presents the conclusions and future work.

II. RELATED WORK

From the provider perspective, the pre-contract stage starts with the reception of the customer request, and finishes when the customer indicates whether or not it accepts the supplier’s proposal [5, 6]. The activities performed during this stage play an important role in the sustainability of software companies; particularly, in micro and medium-sized organizations [7]. The pre-contract represents an opportunity for providers to obtain new contracts, but also a risk because the acceptance of the project bids is not ensured. Proposal accuracy is highly important in this stage, since delivering underestimated bids will negatively affect the sustainability of the company [8].

Fig. 1 shows a summary of the main pre-contract processes reported in the literature. The process labeled as “A”, proposed by Happio and Ahonen [9], indicates the effort estimation and activity planning as the most relevant ones to carry out during the pre-contract stage.
Another process was proposed by the norm “ISO/IEC/IEEE 12207:2017(E): Systems and Software Engineering – Software Life Cycle Processes” [10] (labeled as “B”). It defines a set of activities to conduct part of a supply chain process; the first two activities correspond to the pre-contract stage, and their goal is to identify the product or service that better meets the requirements of the acquirer.

In these two first activities we can find fine-grained tasks like the problem and solution discovery, the proposal preparation and bidding, and the proposal negotiation and adjustment. However, there is not detailed information that allows practitioners to carry out those activities.

Savolainen et al. [5] identify, as other researchers [8, 9], major activities in the pre-contract stage: search, preparation, bidding and negotiation. In the first phase (search), the problem or opportunity to address and the context in which it occurs are identified. Then, the provider analyzes the technical feasibility of the project, and its potential profitability. Based on that, it decides whether or not to prepare a project proposal for the customer.

If it does, during the preparation phase the provider interacts with the customer to define the goals and scope of the product to be developed. It includes the main functionality of the product and also its limits. Considering such a definition, the provider generates a project proposal that includes the project schedule, deliverables, costs and the product scope (bidding).

Finally, some aspects of the proposal could be adjusted through a negotiation process between the stakeholders and the provider. As we can see, these four phases involve several collaborative activities between the participants.

There are also some proposals to address this pre-contract process that come from the business domain. For instance, Cooper and Budd [11] present an adaptation to the sales funnel, which lets a company monitor and control its pre-contract processes from a business perspective. Similarly, Söhnen and Albers [12] define six stages of quality for project proposals that help these proposals evolve through the sales funnel.

Although these pre-contract processes give a general guideline for providers, they are difficult to apply in practice without previous training or detailed documentation. Moreover, it is not clear their level of adoption or effectiveness in the industry, since this aspect has not been reported in recent literature. Next section introduces the study performed to explore the pre-contract processes in the observed Chilean software companies.

III. DESCRIPTION OF THE EXPLORATORY STUDY

We used Grounded Theory (GT) to perform this qualitative study, following the guidelines given in [1, 2, 3, 4]. Fig. 2 shows the main steps performed to gather and analyze the information from the participating companies, and then generate a theory (i.e., the PCP model) for the pre-contract process.

The data collection was conducted using theoretical sampling [4]. This approach seeks and collects pertinent data to elaborate and refine the categories that will then be used in the emerging theory [1, p. 96].

In parallel to the data collection we started the data analysis that included the initial and focused coding [4]. As shown in Fig. 2, the data collection and data analysis are complementary, and they should be performed jointly until achieving theoretical saturation.

During the data analysis, the coders wrote memos (i.e., notes, diagrams, and sketches) to obtain the emerging categories and concepts from the interviewees comments; this activity is known as memoing. In parallel, the coders performed constant comparison among the collected data to verify similarities and differences. Thus, it is possible to create new codes that help in the data analysis process.

The data collection and analysis are performed until they do not produce new codes or concepts; i.e., until achieving theoretical saturation. Then, we can synthesize the coded information and generate the emerging theory, as recommended by GT. Next subsections explain the main aspects of this study.

A. Participants

Twenty engineers from 14 Chilean software companies participated in the study. Four of them belonged to micro-companies, five belonged to small companies, six were part of medium-sized enterprises, and the last five were workers from large companies. All of them were knowledgeable of the pre-contract process of small project prospects in their current companies, and had at least 3 years of experience doing this activity. These were part of the inclusion criteria for the participants in this study.

These companies were focused on performing bespoke projects, and develop mainly ad hoc applications for particular business niches. Most of these applications were web and mobile information systems.

B. Data gathering instruments

The data gathering was performed through semi-structured interviews, one per participant according to the guidelines of Wohlin et al. [13, p. 62]. The interviews were conducted using videoconference and the same session structure.

At the beginning of the session, we informed the participants on the goals of the interview and the dynamic to follow. Then, we asked them about the main items of the questionnaire, which included open and closed questions. The sessions were recorded with the participants consent, and lasted 60 minutes on average.
C. Data processing and coding

The records of the sessions were transcribed, obtaining over 125,000 words (272 pages). The data coding was started as soon as the transcribed information was available, as recommended in [1, p. 45]. This activity was performed manually. After processing the coded information, we created the process model that is introduced in the next section.

IV. PRE-PROJECT PROCESS MODEL

Fig. 3 presents the major activities (indicating if those are mandatory or optional), and workflow of the PCP model. Particularly, the green rectangles indicate activities where the provider interacts with the customer (or stakeholders). The solid borders indicate mandatoriness, and the dashed ones, optionality. The hexagons in purple represent activities that are performed only by the providers team. The arrows between activities indicate the workflow; particularly, solid lines show mandatory transitions and dashed lines indicate optional transitions. The numbers into the activities symbols indicate how many interviewees perform them as part of their pre-contract process. The results show a high agreement on it.

A. Prospecting stage

As shown in Fig. 3, the structure of the PCP model involves two major stages, prospection and pre-sale, and both include particular activities. The prospection stage usually starts with a customer request that comes through an email or phone call to the person (usually a secretary) that formally receives this information and delivers it to the pre-selling personnel. Then, the pre-selling people filter the requests and decide whether or not to upgrade them to leads. If the information is not enough to decide, the provider asks the customer to fill a questionnaire or a checklist to get extra data. The answer is considered by the provider as an indicator of the level of interest, realism and urgency of the customer.

The interviewees indicate that the most frequent sources of new requests are references from other customers, and the repurchase intention of current clients.

After a request becomes a lead, the provider performs a first approach to the client and the lead (shown as “customer initial contact” in Fig. 3), in order to analyze if the lead deserves to be upgraded to the category of project prospect.

According to the interviewees, the key information that providers need to gather on project prospects includes: the actual need to be addressed, the available budget and time, the current situation (as-is), the future situation (to-be), the involved technologies, the number of potential users, and references to similar systems. Depending on it, and also other context information, the provider decides if the lead becomes a project prospect.

B. Pre-selling stage

These prospects usually follow a workflow that includes from the problem and solution discovery, until the decision of the customer on the acceptance of the provider’s proposal. Eventually, any counterpart can abandon the process for any reason; e.g., the prospect became not interesting or unfeasible.

The pre-selling stage starts differently depending on if the customer is new or known. Medium-sized or large suppliers usually count on a pre-sale and a Key Account Management (KAM) area for addressing each of these paths. However, smaller companies fusion these areas for affordability reasons.

In the case of new customers, the providers perform a first meeting to know each other (i.e., the discovery meeting), but mainly to present their business services and context, and also to ask for information about the lead. That information allows both parts to take a look at the customer’s problem and opportunities. The next step in the workflow is to discover the solution.

In case of already known customers, the provider skips the discovery meeting and goes directly to solution discovery activity. According to the interviewees, the discovery meeting and the solution discovery have similar goals; i.e., to explore the problem, context and solution, but at different levels and focusing the work on different aspects. The first one explores the problem and context more in detail, while the second one is more focused on the solution exploration. However, the providers consider the three components a pack that is reviewed in every meeting to gain certainty and detail. Keeping these components together allows the providers to better envision when to continue or quit the efforts on a particular project prospect.

The number of solution discovery sessions depends on several aspects, but mainly on the level of uncertainty of the prospect being addressed. Typically, in small prospects the provider performs one or two sessions.

Once finished the solution discovery, the provider estimates the effort required to develop the solution, and establishes the major deliverables and milestones. Usually, the pre-sale and technical personnel perform this activity, although in micro and small software companies these roles are played by the same person. The estimation process usually follows an expert judgment approach.
Then, the same people prepare the proposal. Twelve participants indicated to deliver separated documents; i.e., a technical and a commercial proposal respectively. The rest of the people deliver a single document that includes both aspects.

The proposal delivery usually includes a presentation to the customer. In such a session there is room to clarify business and technical aspects, and also negotiate the budget or scope of the proposal. After the adjustments, the provider repeats the proposal delivery; the process finishes when the proposal is approved, rejected or canceled. Next we explain the roles participating in the activities of the PCP model.

C. Involved roles

Fig. 4 shows the roles that emerged from analyzing the coded information. The role names are convention among those mentioned by the participants. The numbers indicate many participants of this study use such a role as an actor of the process, and also to support what activity of the PCP model. Next, we briefly explain each of them.

Key Account Manager (KAM) is the person in charge of managing opportunities and needs of the current clients of the company. This role is usually present in providers that keep separated the pre-selling and commercial areas.

Pre-sale personnel usually include people in charge of project sales, who also have technical knowledge of the products and services that the company provides. Frequently, these people are knowledgeable of the business domain they must address. In micro and small companies, this responsibility is assumed by the general manager or an owner.

Technical personnel involve engineers that participate in the development of the solutions. Some roles mentioned by the interviewees as part of this umbrella were: technical area manager, solutions architect, project leader, operations team, and development team. The technical personnel participate more actively in the effort estimation (cost, time and resources).

From the customer side, there are no formal roles defined. However, the information coding process allowed us to identify the following informal roles:

Technical personnel includes people working in technical areas in the customer organization; e.g., technical experts, IT managers, and operations engineers.

Product agents are those who have the actual need; they are the problem domain experts or the product owner.

Decision makers are people able to decide whether or not to continue with the project; e.g., the finance manager, commercial manager or company head.

The role that has more prevalence in the discovery meeting from the provider’s side is the pre-sale specialist (who is part of the pre-sale personnel). Then, in the solution discovery sessions
the technical personnel become protagonists jointly with the presale people. The number of people per role participating in the meetings depends on the project prospect size, complexity and uncertainty level. In small project prospects usually participate one person per role.

D. Supporting tools

The participants reported several tools to support data gathering during the prospection and pre-selling stages; Fig. 5 shows a summary of them considering each participant. All interviewees indicated to perform synchronous and asynchronous activities with the customer. Typically, they perform interviews supported by instruments like questionnaires or canvases.

They also reported asynchronous work of the provider, e.g., the development of mockups or informal diagrams, that are then used to validate or refine proposals. The customer also conducts asynchronous activities, e.g., filling questionnaires or completing checklists that helps the provider to better understand the leads or project prospects.

The variety of instruments (and combinations of them), used by the providers suggest that the pre-selling process is informal and ad hoc; i.e., it is conducted in a different way depending on who performs it, or the project context that is being addressed.

The use of instruments (and combinations of them), used during the pre-selling process is informal and ad hoc; i.e., it is conducted in a different way depending on who performs it, or the project context that is being addressed.

V. ANALYZING THE STRUCTURE OF THE PRE-CONTRACT PROCESS

According to the classification proposed by Di Ciccio et al. [15], the PCP model corresponds to a structured process with ad hoc exceptions, since it has a general structure, but the fine-grained activities should consider external events and exceptions. In support of it, one of the participants indicated “the activities that we perform during the pre-selling depend on the customer availability” [P5]. This makes the provider leave the regular practices to deal with the required adaptation.

This trade-off is also present in the PCP model. Provided that we used open coding to classify the information given by the interviewees, a potential threat to validity could come from wrong interpretation of the transcriptions. In order to address this threat, three researchers performed the open coding, and the others validated a transcription. In order to address this threat, three researchers performed the open coding, and the others validated and adjusted them in various discussion sessions.

VI. THREATS TO VALIDITY

During this research we used several mechanisms to mitigate the threats to the validity of results. First of all, we used GT because this midrange substantive theory allows to focus the analysis to the studied contexts [2, 3]. This context also lets us determine inclusion and exclusion criteria for the participants [16]. Moreover, ensuring the anonymity of the participants and the companies helped us find professionals willing to participate in this study and get honest answers to the questions.

The variety of instruments (and combinations of them), used by the providers suggest that the pre-selling process is informal and ad hoc; i.e., it is conducted in a different way depending on who performs it, or the project context that is being addressed.

The variety of instruments (and combinations of them), used during the pre-selling process is informal and ad hoc; i.e., it is conducted in a different way depending on who performs it, or the project context that is being addressed.

The use of GT is also supported by the soundness of this approach, since the evidence of the generated theory is verifiable using the source data [4]. For that reason, we have explained in detail the processes used in the study (Sect. III), presented the model that emerged from the data processed using GT (Sect. IV), and analyzed the results considering the stated working hypothesis (Sect. V). Thus, we show that the proposed theory accomplishes with the regular validation criteria established for GT. For instance, the PCP model is aligned to the underlying software companies.

In order to deal with the threats to the construction validity, we used the interviews as instrument for gathering data from software companies with different sizes. We conducted this data gathering until getting data saturation.

Provided that we used open coding to classify the information given by the interviewees, a potential threat to validity could come from wrong interpretation of the transcriptions. In order to address this threat, three researchers performed the open coding of the interviews. One of them performed the open coding, and the others validated and adjusted them in various discussion sessions.

VII. CONCLUSIONS AND FUTURE WORK

This article presents an exploratory study to better understand the pre-contract process conducted by software companies when they have to deal with small project prospects.
The working hypothesis of this study indicates that “the providers use an ad hoc and implicit process to perform the pre-contract stage”.

Twenty engineers from fourteen Chilean companies participate in the study. We used Grounded Theory to code and process the collected data. The outcome was the Pre-Contract Process (PCP) model that shows the macro-structure of the process followed by these companies, and also its variability. These results are aligned to the working hypothesis.

This study, and particularly the proposed model, contributes to advancing our knowledge in a project stage that is critical for many software companies, but mainly for the micro and small ones. In this sense, the PCP model can be used to inform the design of pre-contract activities in these software companies, but also in computer science and engineering programs. Undergraduate and graduate students, and also practitioners can take advantage of it.

The future work considers extending this study in order to verify the validity of the PCP model in a larger context, including other Latin American countries.

ACKNOWLEDGMENT

The research work of Anelis Pereira-Vale has been funded by ANID - Subdirección de Capital Humano/Doctorado Nacional/2021-21211216, Chile.

REFERENCES

Fortran Code Refactoring Based on MapReduce Programming Model

Wenhui Gai, Junfeng Zhao*, Han Wu
College of Computer Science, Inner Mongolia University
Hohhot, China
1084302654@qq.com, cszfj@imu.edu.cn, 1500141798@qq.com

Abstract—Fortran language has been widely used to solve computation-intensive tasks in science and engineering. Due to the emergence of multi-core architecture, the pursuit of Fortran parallelism has become an important goal in the field of scientific computing. Because of insufficient computing resources and poor scalability of multi-core architecture, the Fortran program after multi-core parallel still cannot adapt to the explosive growth of data. It is a meaningful work to automatically map parallelizable Fortran code to Spark platform. A Fortran code automatic refactoring and unloading scheme for Spark cluster is proposed in this paper, which is an extension of OpenMP unloading model. The refactoring is automatically completed by the compiler during the compilation process, and the unloading work is automatically completed by calling the unloading function library during the program running process. The experimental results show that the scheme can automatically map Fortran code running on the local computer to the Spark cluster, and improve the execution efficiency of the original business.

Keywords—Fortran; MapReduce; Code refactoring; task unloading

I. INTRODUCTION

There are many researches on parallelization of Fortran language, most of which are based on multi-core CPU. The use of co-processors or accelerators is also a widely studied parallel method. CPU and accelerators constitute a heterogeneous accelerated computing system. Compared with the traditional system based on single-machine multi-core CPU, its key advantage is the high performance power consumption ratio achieved by accelerators, which will gradually replace the previous model in many aspects and become the mainstream of the development of parallel technology in the future.

MapReduce has many efficient implementations[1][2], all of which provide application programming interfaces for developers. Although the specific syntax of different APIs is slightly different, they all require developers to encapsulate the logic of computing tasks into map functions and reduce functions. Developers only need to pay attention to the writing of these two functions. The combination of large data center cluster and MapReduce cloud computing programming model provides an opportunity for the cluster to become an accelerator of local programs. Finally, the program performance can be improved through the powerful parallel computing capability of the cluster.

This paper studies MapReduce refactoring of parallel-cycling code in Fortran programs, and proposes an automatic refactoring and unloading scheme of Fortran code for Spark cluster, which makes the cluster become the accelerator of local Fortran programs. The refactoring work is automatically completed by the compiler during the compilation process, and the unloading work is automatically completed by calling the unloading function library during the program running process.

II. RELATED WORK

Fortran language is the best choice for parallel computing[3]. The pursuit of Fortran parallelism has always been one of the important goals in scientific computing. Early developed Fortran parallel programming interfaces include HPF[4] and CoArray Fortran[5]. In recent years, the widely popular parallel programming interfaces are OpenMP[6] and OpenACC[7] based on instruction programming. Several kernel-based programming interfaces, such as CUDA and OpenCL are also available for Fortran parallelization. In order to further improve the development efficiency of parallelization and reduce the errors easily introduced in the refactoring process of manual parallelization, some researchers have proposed algorithms and tools to realize automatic parallelization refactoring[8]. Tinetti F G proposed a parallelization algorithm to convert legacy Fortran serial code into OpenMP parallel code. The algorithm uses advanced algebraic models to describe code conversion and optimization rules, and uses rewriting rule techniques to automatically apply rules in source code. Advanced algebraic models simplify understanding of legacy programs and their transformations, and can support transformations at different levels of abstraction[9]. Other tools automate parallelization by modifying the output of a compilation system or programming framework. Atzeni S et al. modified the object code generation mechanism of the Flang compilation front end to generate CUDA code for the NVPTX CPU-based back end and automatically unloading the computing tasks to the GPU accelerator. The computing advantage of GPU massively parallel architecture is effectively utilized[10].

Existing research has built converters for several SQL declarative languages and integrated MapReduce to support these languages, including Pig Latin/Pig[11-12], SCOPE[13-14], HadoopDB[15], Hive[16], YSart[17], and Jqal[18]. At present, some scholars have proposed some methods and tools for
refactoring programming language into MapReduce code. Li B proposed a tool J2M (Java-to-MapReduce) that translates Java into MapReduce[19]. This tool is similar to the implementation of an editor, but only compiles target loops with special identification. The object code is generated by extracting some of the necessary information from the source code and combining it with a pre-defined MapReduce template, leaving the rest of the source code unchanged. In order to realize code refactoring for memory cloud computing platform, Li B proposed a translator J2S (Java-to-Spark) that generates MapReduce jobs for Spark platform, which can translate three types of Java source code: for-loop, task, mixed loop and task[20]. Ahmad M B S proposed a conversion tool Casper, which can convert serial Java programs into Spark-MapReduce jobs[21]. In recent years, there are also researches on Fortran language refactoring of MapReduce. Wotruch R proposed OpenMR, a programming model that refactors Fortran, C and C++ source code into MapReduce code[22]. This programming model maps loop iterations to working nodes in a Hadoop cluster based on OpenMP parallel compilation instructions customized in the source code. The compiler can generate map functions and reduce functions required by Hadoop at compile time. While this approach is supported by a set of proof-of-concept, code generation is done manually and there is no comprehensive assessment of the performance overhead of the programming model.

Through the comparison and analysis of relevant research status, it can be found that there are still some problems in the current research. First of all, most existing researches on MapReduce programming model refactoring focus on the transformation from SQL-like queries to MapReduce. The research of refactoring from high-level programming language to MapReduce, most of which focus on object-oriented language such as Java. Secondly, existing research projects require manual participation, and there are still deficiencies in automation. In addition, although the research methods of multi-core parallel refactoring of Fortran programs have been relatively mature in the academia and industry, the refactoring methods of MapReduce programming model are still insufficient, and Fortran programs cannot effectively utilize the powerful parallel computing capability of clusters. The research methods of multi-core parallel refactoring of Fortran programs have been relatively mature, but the refactoring methods of MapReduce programming model are still insufficient. Fortran programs cannot effectively utilize the powerful parallel computing capability of clusters. Fortran, an important language in the field of parallelism, a comprehensive refactoring method is needed to effectively utilize the computing power of clusters. Therefore, this paper studies the refactoring and unloading of Fortran program to MapReduce model.

III. DESIGN OF REFACTORIZING AND UNLOADING SCHEME

In this paper, a refactoring and unloading scheme of Fortran code is designed based on OpenMP unloading model of LLVM compilation system. It is oriented to Fortran source code embedded with OpenMP offload instruction and OpenMP parallel loop instruction, aiming to realize circular parallelization of Fortran code in distributed environment. In this scheme, Spark cluster is selected as the target device because it features fast execution, low computing latency, and high data interaction. According to the analysis of OpenMP unloading model, the data flow diagram of this scheme is shown in the figure 1.

![Figure 1. Data flow diagram](image)

First, the OpenMP target instruction and parallel do instruction are inserted into the Fortran serial source code to guide refactoring and unloading. The source code needs to be correct and only handles Fortran programs that run properly on the CPU, not other types of code. The program embedded with the OpenMP instruction is then read by Flang[23], the Fortran front-end compiler of the LLVM compilation system, which constructs an Abstract Syntax Tree (AST) based on the syntax information of the program. After the whole abstract syntax tree is constructed, the static analysis method of traversing abstract syntax tree is selected for semantic analysis. The AST Consumer and RecursiveASTVisitor interfaces provided by Flang are used to access the abstract syntax tree, and depth-first traversal is performed on the abstract syntax tree. During the traversal, semantic analysis is completed, including the address stored in the variable, the scope of the variable, the name of the variable, the type of the variable and information about the loop. The OpenMP target node of the abstract syntax tree contains all information about the unloading data and the loop to be refactored. The information in the OpenMP target node is saved in the defined data structure to provide information support for the subsequent automatic refactoring and unloading module.

A. Refactoring Scheme Design

For map function refactoring, firstly build the map function's parameters. The first argument is "index", which indicates the lower index limit of the loop in the map function. The second argument is "bound", which indicates the upper bound of the loop index in the map function. The third and subsequent parameters are variables associated with the map instruction, including the map to clause and map from clause. Parameter values are assigned by the Spark driver node. The body of the map function is then built. First the scheme need to extract the loop body in parallel do exactly as it is, and then body link the loop into a new loop that is a child of the original loop with an index upper bound on the first parameter "index" and a lower bound on the second parameter "bound".

For the refactoring of reduce function, first build the parameters of reduce function. The function parameters are the variables in the two reduction clauses, namely a0 and a1. The parameter values are the partial calculation results of the
variable a returned by the map function, passed by the driver node. Then build the body of the function. The body performs the operator specified in the reduction clause for both arguments and assigns the result to the second argument.

This paper uses a Fortran compiler for code refactoring, so the resulting map and reduce functions are machine code for the Fortran language. To enable Fortran functions to be executed in the Spark cluster environment, this paper uses Java Native Interface (JNI) to encapsulate map functions and reduce functions. This needs to be converted to JNI native functions according to JNI naming conventions. The Spark cluster also needs to generate a Spark application that describes cluster jobs to trigger cluster jobs. Based on the analysis of Spark execution mode, this paper designs the Spark job application template. During refactoring, the specified information is inserted into the template to generate Spark job applications for computing tasks.

**B. Unloading Scheme Design**

The automatic unloading module relies on the flexible implementation of the OpenMP unloading model.

The data processing part mainly deals with the associated variables of map instruction. The OpenMP unloading model is used to process data mapping of variables associated with map instruction. Meanwhile, in order to reduce the overhead of moving data across the Internet, this paper extends the use of map instruction clause and implements distributed data partitioning. Based on the Spark cluster architecture, this paper designs an unloading function library that can directly interact with the Spark cluster and has the unloading function in the function interface provided by the OpenMP unloading model. Relying on the OpenMP unloading model, the compiler replaces the area of code associated with the target directive with a call to the unloading library. Then run the program to realize the automatic unloading of calculation tasks.

**IV. IMPLEMENTATION OF REFACTORING AND UNLOADING SCHEME**

Based on the detailed design in Chapter III, this chapter details the implementation of the refactoring and unloading solution to support the refactoring and unloading process. This paper is implemented in the LLVM compilation system. The following introduces the overall implementation method.

To achieve the extensibility of the OpenMP unloading model, the LLVM compilation system breaks the implementation of the model into different components, including a compiler that generates object code, an unloading wrapper library that is independent of the target device, and an unloading plugin that is specific to the target device. This paper extends the LLVM compiler to generate object code for the Spark cluster and the libomptarget runtime unloading library, where the unloading plugin is implemented to unloading computing tasks into the cluster. The implementation name of the defined scheme is FCLOUD (Fortran-Cloud), which is composed of LLVM compiler, unloading wrapper library, and cluster plugin. The overall implementation is shown in the figure 2.

The LLVM compiler implements the analysis and compilation of the input source program, during which three types of code are generated. The first is the map and reduce functions running on the cluster, which are contained in JNI region. The second is the code that runs on the mainframe, which is contained in program main, embedded in the same binary in ELF format. The third code is the Spark job application code compiled into a JAR file. When a job is submitted to the cluster, the driver node runs the Spark job application and distributes cyclic tasks among working nodes. Then the working node runs the map and reduce functions written in Fortran language through the JNI interface.

The unloading wrapper library component was designed by Jacob, in the LLVM compilation system to achieve the scalability of OpenMP unloading. The main tasks were to detect the available target devices, create the device data environment, perform the correct unloading function according to the device type, etc. When an unloading area "target" is synthesized, the LLVM compiler generates a set of calls to the unloading wrapper library, regardless of the target device.

Cluster unloading plugin is a component specific to the target device in the libomptarget runtime unloading library. Currently, there are GPUs for GPU and DSPs for DSP. The cluster plugin implemented in this paper can directly interact with the cluster according to the architecture of the cluster, and provide services such as data transmission and triggering the cluster to execute jobs. Because the cluster device is not set up on the local computer, it cannot be automatically detected. Users need to provide identification or authentication information to allow the current application to connect to the cloud service implementation for unloading. The cluster plugin reads the configuration file at runtime to set up the cluster device correctly. Therefore, you need to deploy the cluster and set up the configuration file before running the application. The implementation of LLVM compiler component corresponding information collection module and code refactoring module, unloading packaging library and cluster plugin corresponding unloading module.
V. EXPERIMENTS

A. Experimental Environment and Test Procedure

The local computer is a laptop with 8 GB of memory. The target cluster consists of a private Spark cluster with a driver node and 16 working nodes. Use the libssh API to implement SSH/SFTP communication between the local computer and the target cluster. Table I describes the configuration parameters of each node in the cluster.

<table>
<thead>
<tr>
<th>Configuration name</th>
<th>Configuration description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>Ubuntu14.04</td>
</tr>
<tr>
<td>CPU processor</td>
<td>AMD Opteron(TM) Processor 2376 CPU @ 2.3GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>16GB</td>
</tr>
<tr>
<td>Spark version</td>
<td>2.3.1</td>
</tr>
<tr>
<td>JDK version</td>
<td>1.8.0 171</td>
</tr>
<tr>
<td>Kernel number</td>
<td>8</td>
</tr>
</tbody>
</table>

In this paper, PolyBench/Fortran suite[24] is used as the benchmark test assembly, from which programs that support the OpenMP target structure and use the typical do loop implementation are selected: SYRK, Mat-mul, 2MM, 3MM, Collinear-list. All data sets used in the experiment are composed of randomly generated single precision numbers, and the matrix used is expanded to \(16000 \times 16000\) (about 1GB). Taking the Mat-mul serial program as an example, the refactoring and unloading process based on FCloud is as follows: First, write the Spark cluster configuration file, and then insert instructions in the Mat-mul serial source code according to the standard description mode defined by OPenMP. During code compilation, FCloud completes Spark-MapReduce refactoring of parallel cyclic code and generates calls to unloading library functions. By running logs to learn about the unloading process and Spark job execution. After the running is complete, the log displays the execution time and other information.

B. Feasibility Analysis

This section verifies whether FCloud can complete the refactoring and unloading by comparing the running results of FCloud and the original serial program. First, use FCloud to test five applications. Table II shows the number of loops contained in the program, the number of nested loops in the program, and the corresponding completion results.

<table>
<thead>
<tr>
<th>Program</th>
<th>Loop nesting level</th>
<th>Number of cycles</th>
<th>Whether it can be compiled and run normally</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYRK</td>
<td>2</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>Mat-mul</td>
<td>3</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>2MM</td>
<td>3</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>3MM</td>
<td>3</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>Collinear-list</td>
<td>4</td>
<td>3</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This paper compares the execution results of this scheme and the original serial program in the same data set to verify whether the external behavior of the program has changed, and then to verify whether this scheme has effectively completed the refactoring and unloading. It is mainly to check whether the serial program execution results are consistent with the program execution results of this scheme through traversal, and the verification results are consistent.

C. Performance Analysis

To test the performance of this scheme, this paper compares the clustered distributed parallelism of FCloud with the parallelization method popular in Fortran, OpenMP single-machine multithreaded parallelism. The main way is to compare the run speed of the two programs with that of the serial program. Run speed is the most important indicator to test the performance of the parallel scheme.

Since a single machine only has a maximum of 8 cores, single-machine multithreaded OpenMP experiments were only tested on 2, 4, 6 and 8 threads. As shown in the running results of OpenMP in Figure 3 (a), the acceleration of other programs is close to linear except that Collinear-list program accelerates 1.5 times on 8 inner cores. Figure 3 (b) shows that the overall speed-up of all tested programs on FCloud tends to increase with the number of cores, with 3MM programs getting up to 58 times speed-up at 128 cores. At the same time, it can be seen that even the Collinear-list program, which performs the worst in single-machine multithreaded OpenMP, can achieve 12.5 times speed-up with 128 cores.

Through the above analysis, it can be concluded that this scheme can effectively improve the operation efficiency of the program. Compared with multithreaded OpenMP technology whose parallel capability is limited by the number of single cores, although the acceleration achieved in this scheme is not linear, it will increase with the increase of the number of cores. As the number of cores increases, the 3MM program with the maximum computational complexity gains the largest execution acceleration compared with other tasks with lower computational complexity, indicating that the program with more complex computation can gain greater acceleration through this scheme as the number of cores increases.
D. Performance Overhead Analysis

The results of the experiment show that the acceleration of FCloud does not increase linearly with the increase of the number of cores, and in the case of 8 cores, the running acceleration is lower than that of single-machine multithreaded OpenMP technology, which is caused by the performance overhead generated by FCloud when running the program. This section mainly analysis its performance overhead to verify whether the scheme has a good performance power consumption ratio.

Figure 4 shows the running time of FCloud and multithreaded OpenMP on a single node.

As the number of cores increases, the computation time of FCloud decreases, while the communication time and Spark spending remain roughly the same. Comparing the run time of FCloud on 8 cores to the run time of single-machine multithreaded OpenMP also shows a smaller performance overhead. When computing time alone is considered, FCloud has 4.3% more overhead than multithreaded OpenMP, which proves how efficient JNI is at running native functions. When the overhead of Spark clusters is added, FCloud is 6.7% more expensive than multithreaded OpenMP, demonstrating the Spark platform's superior parallelism performance even in the driver node-worker node execution architecture. When the communication overhead between the local computer and the cluster is added, the total running time of the FCloud is 7.6%
higher than that of the single multithreaded OpenMP, which indicates that the overhead of transferring data between the local computer and the Spark cluster is limited in the small number of clusters, even without great computing power.

Based on the above analysis, it can be concluded that the overhead of data transmission and Spark scheduling is stable and limited. Compared with the significantly reduced computing time, the overhead caused by the performance is small, and the scheme has a good performance-power ratio, which further proves the effectiveness of the scheme.

Experimental results show that this scheme can complete the parallel refactoring and unloading of Fortran programs, and has good performance when dealing with large data sets and complex computing applications. Meanwhile, the overhead of scheduling and data transmission within Spark is limited, and JNI runs local functions efficiently, which proves that this solution has a good performance-power ratio.

VI. CONCLUSION

In this paper, a Fortran code automatic refactoring and unloading scheme for Spark cluster is proposed. A parallel refactoring method is proposed based on the parallel mode of MapReduce programming model. The OpenMP unloading model is used to automatically uninstall the computing tasks on the local computer to the Spark cluster. The experimental results show that the scheme is feasible, effective and has good performance and power consumption ratio. The application of the refactoring and unloading scheme proposed in this paper can help to improve the efficiency of specific Fortran programs. For future work, it would be interesting to cover more Fortran code and reduce performance costs.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (61962039).

REFERENCES

Abstract—To port the Linux distributions to a new Instruction Set Architecture (ISA), developers have to rebuild the software packages of the distributions. The complex dependencies of the software packages bring a great challenge. It is important to understand and properly handle the dependencies. We selected Fedora, a typical Linux distribution, and studied the dependencies within the software repositories of aarch64 and x86_64 architecture. We proposed a package dependency network framework to study the roles played by different packages. We obtained three network dependency patterns and proposed the corresponding division strategies which help developer build the source packages in parallel. Our study reveals that the key packages located at the root of multiple dependency chains significantly impact the division of the network, and their builds should be prioritized. Meanwhile, some packages with external dependencies can be temporarily masked to make a sub-network independent. Furthermore, the network dependency patterns are also observed in Fedora 33 riscv64 and OpenEuler riscv64. Our findings can help researchers have a better knowledge of Linux distribution dependency network and help practitioners conduct efficient package builds.

Keywords—software repository, build dependency, software porting, dependency network

I. INTRODUCTION

With the advent of new ISAs such as ARM and RISC-V, developers of Linux distributions are seeking to port their distributions to these new architectures and establish corresponding software ecosystems [1]. However, porting a distribution to a new ISA is far from straightforward. Firstly, building software from source packages to binaries requires the support of other software known as build dependencies (reused software). The building process of a porting has to be started from scratch and conducted in dependency order. Secondly, the complex dependencies between source packages can be overwhelming for developers. Furthermore, porting a distribution to a new architecture involves rebuilding a large number of source packages, which is a time-consuming task.

Ye [2] focuses on the order of source packages building, using topological sorting to give the best building order. In this paper, we conducted an empirical study of the dependency relationship between source packages in Linux distribution software repositories. We crawled all packages, including source and binary packages, from 4 Fedora repositories for X86 (x86_64) and ARM (aarch64) architectures and 2 repositories of OpenEuler and Fedora 33 for the emerging RISC-V (riscv64) architecture. We built a package network for each repository based on the dependency relationships between source packages, and studied the networks for efficient porting.

We analyzed the connections of packages in the network to check whether the network can be divided into sub-networks for parallel building. Our results show that source packages can be aggregated into clusters, with sparser connections between the clusters than within the clusters. It implies that it is possible to fragment the network into sub-networks. We also obtained some insights of the role that different source packages play in the network. We found that the successful builds of the key packages belonging to multiple dependency chains are important for the builds of the dependent packages. These key packages have a significant impact on parallel building. Finally, according to the prior findings, we attempted to divide the dependency network into sub-networks and identified three dependency patterns and the associated division strategies.

The main contributions of this paper are as follows: (1) We built a package dependency network framework of Linux distributions; (2) We obtained three dependency patterns to help developers improve package build efficiency.

The remaining paper is organized as follows. Section 2 introduces our motivation and research questions. Section 3 presents the dataset. Section 4 illustrates our method and reports the results. Section 5 introduces related work. Section 6 concludes.

II. MOTIVATION AND RESEARCH QUESTIONS

Porting a Linux distribution to new ISAs often requires rebuilding all of its software packages from scratch with unsupported dependencies. As depicted in Fig. 1, since the build of the source package accountservice [3] relies on the output of building gobject-introspection, the build of the former cannot be started until the successful build of the latter. This is a simple example of a dependency chain of source packages.

The numerous dependency chains form a large network, which may be split into smaller sub-networks for parallel builds. As shown in Fig. 2, after the successful builds of the zlib, make, and gcc, the network becomes two independent sub-networks which have 4 and 8 packages separately. They can be assigned to different teams for parallel builds.

To explore the division of the dependency network, we raise the following two research questions: RQ1: What are the characteristics of the dependency network? RQ2: How can a dependency network be divided into independent sub-networks?
III. DATASET

The dependency network of a repository can be defined as a directed network $G(V, E)$. $V = \{v_1, v_2, ..., v_i\}$ is the set of source packages in the repository, and $E = \{e_i, e_2, ..., e_M\}$ is the set of dependency edges, where $e_i = \{v_i, v_j\}$ $(i = 1, 2, ..., M)$. Source packages can be grouped into a cluster, i.e., a sub-network $G_{sub}(V_{sub}, E_{sub})$, where $V_{sub} \subseteq V$, $E_{sub} \subseteq E$. To examine the relationships between the sub-networks, some relevant external source packages are added into the original sub-network, which forms the merged sub-network.

To collect the edges between source packages in the dependency network, the following relations were extracted: the generation relation from an SRPM (source package) to an RPM (binary package), the providing relation from RPM to binary modules, and the build dependency relation from the SRPM to binary modules. These information can be acquired through the rpmspec [4] and rpm [5] tool. Take the dependency chain in Fig.1 as an example. Firstly, the rpmspec is used to parse the SPEC file of `accounts-service` to obtain its build dependencies using rpmspec. Next, the rpm is utilized to retrieve the list of modules provided by `gobject-introspection-devel`, which includes `giscanner`. Finally, the SPEC file of `gobject-introspection` is parsed again to obtain its build dependencies using rpmspec. In consequence, the relations of the source packages from `accounts-service` to `gobject-introspection` are obtained.

### III. METHODOLOGY & RESULTS

#### A. Ans. to RQ1: Attributes of Source Packages Network

To answer the two research questions, we crawled all packages in the 6 repositories mentioned in Section 1, the results of the network construction are presented in Table 1. Our initial analysis reveals that over 97% of the source packages are interconnected in the largest connected component. The remaining fragmented packages have minimal effect on network analysis. Thus, they are eliminated.

#### B. Ans. to RQ2: Scale-Free, Small-World & AdaptiveNetworks

### IV. METHODOLOGY & RESULTS

#### A. Ans. to RQ1: Attributes of Source Packages Network

Dividing a network into sub-networks requires identification of community structures. The sparsity between communities can reduce the complexity of the division. Meanwhile, the efficiency of the building process may be impacted by the connections between the packages, as packages with a large number of dependencies may block the builds of many others. Therefore, we analyzed the internal structure and the degree distribution (dependencies distribution) of the dependency network.

**Small-World:** The average shortest path length (ASPL) [6] represents the efficiency of dependency transferred in the network, and the average clustering coefficient (ACC) [7] is the probability that packages gather into a cluster. We compare ASPL and ACC of Fedora dependency network (FedoraGraph) with a random directed network (RandomGraph) of the same scale. Taking Fedora 34 x86_64 as an example, as shown in Fig. 3, the FedoraGraph has a shorter ASPL (0.42) and a higher ACC (0.30), which indicates that the network has a small-world attribute [8], i.e., it is highly connected and aggregated. Nodes in the network tend to gather into clusters, and the relationships between the clusters are sparser than those within the clusters. This finding suggests that the dependency network can be divided into sub-networks.

#### B. Ans. to RQ2: Scale-Free, Small-World & AdaptiveNetworks

**Scale-Free:** Fig. 4 shows the log-log plots of the degree distribution, where the horizontal axis represents the in/out degree of source packages and the vertical axis is the corresponding distribution. Our fitting results show that the degree distribution follows a power-law distribution ($\gamma = 1.836$, and $\gamma_{out} = 2.099$), indicating the scale-free attribute [9] of the network. The majority of the packages in the repository occupy upstream positions in the dependency chain and do not provide support for the builds of other packages. A large number of dependencies are concentrated in a minority of the packages, which we refer to as key packages. Consequently, the successful builds of key packages can significantly reduce the overall number of unsupported dependencies for the packages of a Linux distribution.
The network is divided into 14 sub-networks in a sub-network. The cases where the dependencies of most packages are within the sub-network itself. Developers can build this kind of sub-networks in advance. Fig. 6(a) depicts the merged sub-network A (A1-A6) to B (B7-B8). The red nodes (A1 and A4) are the key packages of sub-network A. A5 and A6 depend on B7 and B8 of sub-network B. In this pattern, the union of A5 and A6 does not support other source packages. Whether these nodes are successfully built or not has little effect on most of the source packages in the sub-network. Developers can put off the build of A5 and A6 when scheduling.

Core dependency (CD): The external dependencies of a sub-network are concentrated on the upstream packages (i.e., nodes with 0 in-degree), a sub-network has the partial dependency. In this pattern, although the sub-network still depends on packages from the external sub-network, the dependencies of most packages are within the sub-network itself. Developers can build this kind of sub-networks in advance. Fig. 6(a) depicts the merged sub-network A (A1-A6) to B (B7-B8). The red nodes (A1 and A4) are the key packages of sub-network A. A5 and A6 depend on B7 and B8 of sub-network B. In this pattern, the union of A5 and A6 does not support other source packages. Whether these nodes are successfully built or not has little effect on most of the source packages in the sub-network. Developers can put off the build of A5 and A6 when scheduling.

General dependency (GD): The dependencies of most packages of a sub-network are a few external packages. The sub-network of this pattern cannot be easily separated from the external. As shown in Fig. 6(c), A2, A3, and A5 in sub-network A depend on B6 and B7 in sub-network B. Although the two key packages A1 and A4 do not form core dependency, the sub-network cannot be divided due to the dependency of B6 and B7. Developers should complete the builds of a few external source packages to make the sub-network only have partial dependency.

Based on the prior observations, we proposed two heuristics for network division: (1) Key packages in the sub-network must be made independent to ensure the builds of all related dependency chains. (2) The sub-network does not need to be completely independent of external dependencies. Developers can work on a sub-network when the majority of its packages can be built. With these heuristics, we identified three patterns of sub-network dependency as depicted in Fig. 6.

**Partial dependency (PD):** When the external dependencies are concentrated on the upstream packages (i.e., nodes with 0 in-degree), a sub-network has the partial dependency. In this pattern, although the sub-network still depends on packages from the external sub-network, the dependencies of most packages are within the sub-network itself. Developers can build this kind of sub-networks in advance. Fig. 6(a) depicts the merged sub-network A (A1-A6) to B (B7-B8). The red nodes (A1 and A4) are the key packages of sub-network A. A5 and A6 depend on B7 and B8 of sub-network B. In this pattern, the union of A5 and A6 does not support other source packages. Whether these nodes are successfully built or not has little effect on most of the source packages in the sub-network. Developers can put off the build of A5 and A6 when scheduling.

**Core dependency (CD):** The external dependencies of a sub-network are concentrated on the upstream packages (i.e., nodes with 0 in-degree), a sub-network has the partial dependency. In this pattern, although the sub-network still depends on packages from the external sub-network, the dependencies of most packages are within the sub-network itself. Developers can build this kind of sub-networks in advance. Fig. 6(a) depicts the merged sub-network A (A1-A6) to B (B7-B8). The red nodes (A1 and A4) are the key packages of sub-network A. A5 and A6 depend on B7 and B8 of sub-network B. In this pattern, the union of A5 and A6 does not support other source packages. Whether these nodes are successfully built or not has little effect on most of the source packages in the sub-network. Developers can put off the build of A5 and A6 when scheduling.

**General dependency (GD):** The dependencies of most packages of a sub-network are a few external packages. The sub-network of this pattern cannot be easily separated from the external. As shown in Fig. 6(c), A2, A3, and A5 in sub-network A depend on B6 and B7 in sub-network B. Although the two key packages A1 and A4 do not form core dependency, the sub-network cannot be divided due to the dependency of B6 and B7. Developers should complete the builds of a few external source packages to make the sub-network only have partial dependency.

Based on the prior observations, we proposed two heuristics for network division: (1) Key packages in the sub-network must be made independent to ensure the builds of all related dependency chains. (2) The sub-network does not need to be completely independent of external dependencies. Developers can work on a sub-network when the majority of its packages can be built. With these heuristics, we identified three patterns of sub-network dependency as depicted in Fig. 6.

**Partial dependency (PD):** When the external dependencies are concentrated on the upstream packages (i.e., nodes with 0 in-degree), a sub-network has the partial dependency. In this pattern, although the sub-network still depends on packages from the external sub-network, the dependencies of most packages are within the sub-network itself. Developers can build this kind of sub-networks in advance. Fig. 6(a) depicts the merged sub-network A (A1-A6) to B (B7-B8). The red nodes (A1 and A4) are the key packages of sub-network A. A5 and A6 depend on B7 and B8 of sub-network B. In this pattern, the union of A5 and A6 does not support other source packages. Whether these nodes are successfully built or not has little effect on most of the source packages in the sub-network. Developers can put off the build of A5 and A6 when scheduling.

**Core dependency (CD):** The external dependencies of a sub-network are concentrated on the upstream packages (i.e., nodes with 0 in-degree), a sub-network has the partial dependency. In this pattern, although the sub-network still depends on packages from the external sub-network, the dependencies of most packages are within the sub-network itself. Developers can build this kind of sub-networks in advance. Fig. 6(a) depicts the merged sub-network A (A1-A6) to B (B7-B8). The red nodes (A1 and A4) are the key packages of sub-network A. A5 and A6 depend on B7 and B8 of sub-network B. In this pattern, the union of A5 and A6 does not support other source packages. Whether these nodes are successfully built or not has little effect on most of the source packages in the sub-network. Developers can put off the build of A5 and A6 when scheduling.

**General dependency (GD):** The dependencies of most packages of a sub-network are a few external packages. The sub-network of this pattern cannot be easily separated from the external. As shown in Fig. 6(c), A2, A3, and A5 in sub-network A depend on B6 and B7 in sub-network B. Although the two key packages A1 and A4 do not form core dependency, the sub-network cannot be divided due to the dependency of B6 and B7. Developers should complete the builds of a few external source packages to make the sub-network only have partial dependency.

We also observe these patterns in the sub-networks of Fedora 33 riscv64 and OpenEuler riscv64. As demonstrated in Table 2, all the sub-networks, excluding the Mix sub-network, match one of the patterns and can be divided in the ways discussed above.
The majority are unable to work without these dependencies. Similarly, Gou et al. and a few binary packages bear most of the dependencies, and the empirical study on seven software ecosystems and found that changes of resilience during software evolution by modeling Gao et al. [17, 18] modeled and studied different software of Linux distributions. Yao et al. [6] analyzed defects and provides a way to sort the build order of source packages knowledge, only Ye [2] paid attention to the build dependency Jar files. Li et al. and Prana et al. [15, 16] studied dependency tools [11-14] have been proposed to address dependency conflicts and redundant dependencies in Python programs and Python multiple dependency chains and can block the build process potential to divide the network into sub-networks. The key conduct parallel builds of source packages.

For Linux distributions, developers must rebuild the software of many other packages. We identified three patterns of sub-strategies of division. Our work can inspire further research in independence of the sub-networks, as they act as the root of many other packages. We identified three patterns of sub-network dependency and provided the corresponding strategies of division. Our work can inspire further research on the dependency network and help developers efficiently conduct parallel builds of source packages.

V. RELATED WORK

Software reuse often leads to dependency problems, and most of the studies focused on runtime dependencies. Several tools [11-14] have been proposed to address dependency conflicts and redundant dependencies in Python programs and Jar files. Li et al. and Prana et al. [15, 16] studied dependency conflicts and dependency vulnerability respectively. To our knowledge, only Ye [2] paid attention to the build dependency and provides a way to sort the build order of source packages of Linux distributions. Yao et al. [6] analyzed defects and changes of resilience during software evolution by modeling functions in the Android OS kernel. Similarly, Gou et al. and Gao et al. [17, 18] modeled and studied different software systems at the function level. Decan et al. [19] conducted an empirical study on seven software ecosystems and found that a few binary packages bear most of the dependencies, and the majority are unable to work without these dependencies.

VI. CONCLUSIONS

One of the biggest challenges faced by emerging ISAs is building the corresponding software ecosystem from scratch. For Linux distributions, developers must rebuild the software repository to support the emerging ISAs, where the build dependencies are very complex. In this paper, we conducted an empirical study of multiple repositories to explore whether the dependency network can be divided into sub-networks. Our findings reveal that the dependency network presents scale-free and small-world attributes. Source packages within the network tend to gather into clusters, which offers the potential to divide the network into sub-networks. The key packages in the network significantly impact the independence of the sub-networks, as they act as the root of multiple dependency chains and can block the build process of many other packages. We identified three patterns of sub-network dependency and provided the corresponding strategies of division. Our work can inspire further research on the dependency network and help developers efficiently conduct parallel builds of source packages.

TABLE II. THE DIVISION RESULT OF SUB-NETWORKS IN FEDORA 33 RISCV64 AND OPENEULER RISCV64

<table>
<thead>
<tr>
<th>Repository</th>
<th>Sub-Network</th>
<th>PD</th>
<th>CD</th>
<th>GD</th>
<th>Divisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fedora 33</td>
<td>Python</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Perl</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Drupal7</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Emacs</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Erlang</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Gap</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Globus</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Golang</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Hyperbion</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Java</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Mix</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>NodeJs</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Ocaml</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Perl</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>PHP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Python</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Ruby</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Rust</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>

REFERENCES

EasyPip: Detect and Fix Dependency Problems in Python Dependency Declaration Files

Shuo Li
Institute of Software Chinese Academy of Sciences, University of Chinese Academy of Sciences
Beijing, China
lishuo19@otcaix.iscas.ac.cn

Jie Liu*
State Key Lab of Computer Sciences at ISCAS, University of Chinese Academy of Sciences Nanjing, China
ljie@otcaix.iscas.ac.cn

HaoXiang Tian
Institute of Software Chinese Academy of Sciences, University of Chinese Academy of Sciences Beijing, China
tianhaoxiang20@otcaix.iscas.ac.cn

Shuai Wang*
Institute of Software Chinese Academy of Sciences, University of Chinese Academy of Sciences Beijing, China
wangshuai@otcaix.iscas.ac.cn

Wei Chen
State Key Lab of Computer Sciences at ISCAS, University of Chinese Academy of Sciences Nanjing, China
wchen@otcaix.iscas.ac.cn

Liangyi Kang, Dan Ye
Institute of Software Chinese Academy of Sciences Beijing, China
{kangliangyi15,yedan}@otcaix.iscas.ac.cn

Abstract

Environment configuration is the basis for software reuse, enabling developers to reuse specific functions. However, the lack of uniform practice in dependency declaration specifications of Python projects can cause problems for developers trying to install third-party libraries. Existing package management tools are often inadequate to help fix these problems. Fixing these errors requires expensive hours and domain knowledge for developers.

To help address related problems, some studies focus on well-maintained and popular Python projects about dependency conflict problems caused by PIP’s installation rules. However, many projects in the wild are outside of this scope. We carefully investigate 110 issues in 110 projects in the wild. Based on the comprehensive study, we design and implement EasyPip to automatically detect and fix problems in Python dependency declaration files. Different from existing tools, EasyPip can locate conflicting dependencies without trying to install dependencies, and generate fixing solutions with the least modification to the original files. We evaluate EasyPip on the collected dataset which shows that EasyPip outperforms two state-of-the-art tools and can effectively detect problems in 91.04% Python dependency declaration files and generate feasible fixing solutions for 65.67% of them.

Index Terms—Software reuse, dependency declaration file, dependency error fix, Python

I. Introduction

Python is one of the most popular programming languages, but its dependency declaration lacks uniform specifications, which may lead to unknown problems. We call the third-party library installation problems caused by the dependency declaration files instead of package management tools dependency declaration issues. The issues can be seen in real projects, such as Im2Vec [1] whose requirements.txt file specifies conflicting dependencies, leading to dependency declaration issues.
The dependency declaration file should contain all third-party libraries used in the project to ensure successful installation. PIP and Conda can export the dependency declarations of the installed third-party libraries in the environment of the project to help reuse Python projects. However, the compatibility between direct dependencies and transitive dependencies is ignored. Recent techniques like PyEgo [2] infer compatible environment dependencies’ versions for Python code snippets, but couldn’t identify potential conflicts that weren’t in their knowledge database. And they ignore that the original dependency declaration file provided by the project developer is closer to the environment configuration that the developer expects. And developers will likely perform smaller updates to mitigate the impact of breaking changes. So, the fewer modification to the original dependency declaration file, the better to successfully build the project and achieve the expected results. Therefore, detecting and fixing dependency declaration issues with less modification is helpful for developers to reuse Python projects.

Wang et.al. [3], [4] investigate dependency conflict issues with the legacy and new dependency-resolving strategies adopted in PIP. However, the studies focus on well-maintained projects. There are many projects left in the wild not well-maintained or unpopular. Developers will face more challenges when reusing these projects.

To investigate the characteristics of dependency declaration issues of projects in the wild, we conduct an empirical study on dependency declaration files in Python projects from Github. We collect 1000 issues related to dependency declaration, and among them, 110 dependency declaration issues of 110 Python projects are identified. We thoroughly analyze these issues and conclude manifestation patterns from them. Note that different from the empirical study of Watchman [3], our study mainly focuses on the dependency conflicts caused by dependency declaration files instead of those caused by the installation strategy of PIP.

Based on our study, to address dependency declaration issues and help developers successfully install third-party libraries when reusing Python projects, we design and implement a tool, EasyPip, which can automatically detect and fix dependency declaration issues. Differently from the existing tools (e.g., PIP, SMARTPIP [4]), EasyPip can detect and locate dependency declaration issues without trying to install, and generate feasible fixing solutions with the least modification to the original files. Specifically, EasyPip formulates the detection and fixing of dependency declaration issues as a graph-search problem. To improve the efficiency of detection and location, we introduce the equivalent node to reduce search space. To generate the feasible fixing solutions with the least modification, EasyPip utilizes greedy search to find the compatible versions closest to the original dependency declaration file.

We evaluate EasyPip on the dataset of dependency declaration files with dependency declaration issues from open-source projects. The results show that, EasyPip can effectively detect 91% dependency declaration issues and report the root causes, and generate feasible fixing solutions for 72% of the detected issues. Comparing to the state-of-art techniques for detecting and fixing dependency declaration issues, on the same dataset, EasyPip detects 12% more dependency declaration issues and generates 34% more feasible fixing solutions. For the fixing solutions generated by EasyPip and the selected baselines, we calculated their changes in dependencies and versions to the corresponding original files. The results show that the fixing solutions generated by EasyPip are closer to the original dependency declaration files.

In summary, the main contributions of our work are as follows:

- We conduct the empirical study of dependency declaration issues in Python projects in the wild. Our findings can help further understand the characteristics of dependency installation problems.
- Based on our findings, we design and implement a tool, EasyPip, to help automatically detect and locate dependency declaration issues without installing, and generate feasible fixing solutions with the least modifications to the original dependency declaration file.
- We release the dataset used in our empirical study and evaluation, which can facilitate future research related to Python dependency declaration issues.

II. EMPIRICAL STUDY

A. Data Collection

To collect dependency declaration issues, we follow the data collection practice of empirical studies [5]–[8] on Python developer communities, and collect our data by the following steps: First, we go through issues that are related to dependency declarations of Python projects on GitHub. Next, we manually read the issue reports and comments, to identify whether it is a dependency declaration issue.

1. Collecting Python projects: We search Python projects on GitHub with issues filtered by the keywords: “requirement” “dependency” or “install”, and obtain 1000 issues. Then we filter out the projects without dependency declaration files (requirement.txt and setup.py). The duplicate issues are dropped with the first one left for these keywords can appear at the same time in one issue.

2. Identifying dependency declaration issues: We then go through each issue report and its comments, and further identify dependency declaration issues by several criteria: a. the issue is caused by a dependency declaration file
instead of the user’s incorrect operation and PIP’s installation, b. the corresponding release of the project is available so that the issue can be reproduced. Three of the authors independently check each issue, and then discuss it to reach an agreement. After filtering by the criteria above, there are 110 issues in 110 projects left.

**B. Manifestation Patterns**

We categorize dependency declaration issues into four categories (Pattern A to D) according to their occurrence stages and root causes.

**Manifestation of Pattern A:** The dependency declaration issues of Pattern A manifest as different dependencies required by the project cannot be satisfied at the same time (57/110). As conflicts come from different sources, Pattern A is divided into three sub-categories. **Pattern A.1:** conflicts between direct and transitive dependencies (47/57). **Pattern A.2:** conflicts between transitive dependencies (10/57).

**Manifestation of Pattern B:** The issues of Pattern B are caused by Python interpreters (10/110). Different dependencies in the dependency declaration file require different Python interpreters.

**Manifestation of Pattern C:** The issues of Pattern C are caused by incompatibility with Operating System (13/100). The dependency in the dependency declaration file is incompatible with the OS of the developer who wants to reuse the project.

**Manifestation of Pattern D:** The dependencies of the project and that of its upstream/downstream project, cannot be satisfied at the same time (19/110).

The other issues (11/110) are caused by unavailable dependency (e.g., library not found or not released publicly) declared in the file.

From our study, it can be seen that most dependency declaration issues are caused by conflicts among declared dependencies, transitive dependencies, and Python interpreters.

**III. EasyPip**

To help automatically solve the dependency declaration issues without trying to install dependencies, we design and implement a tool, EasyPip, which can efficiently detect and locate the issues of the two most common patterns (Pattern A and B). EasyPip can also find feasible fixing solutions with minimal modifications to the original dependency declaration file as developers will likely perform smaller updates to mitigate the impact of breaking changes. The workflow of EasyPip is shown as Figure 1.

**Dependency Graph Construction** The input of EasyPip is a dependency declaration file. Firstly, EasyPip parses it to get the third-party libraries and their version constraints. Then EasyPip queries their transitive dependencies from the dependency knowledge database. Based on these dependencies and their relations, EasyPip constructs the dependency graph of these dependencies with Python interpreters.

In the dependency graph, each node represents a Python interpreter or a third-party library with a specific version. The edges contain two types: seq_edge (the undirected edge) and dep_edge (the directed edge). The nodes of different versions in the same third-party library or interpreter are connected by seq_edges. The nodes of different libraries or interpreters with dependency relations are connected by dep_edges. As the latest versions are preferred in Python project development, the nodes of the same library or interpreter are ordered in descending of versions.

For example, there are two dependency declarations in the file: \( B == 5, C \). The dependency relations of \( B \) and \( C \) are shown as Table I and the constructed dependency graph is shown as Figure 2 (Note that to intuitively display the example, the Python interpreters are not shown in the figure). In the dependency graph, for each library, EasyPip denotes the nodes with maximal version and minimal version of corresponding declared version constraints (If there is no declared version constraint on the dependency, the nodes from the oldest version to the newest version of the dependency will all be denoted).

**TABLE I. Dependency relation**

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Dependent package</th>
<th>Dependent version</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>V1-V2</td>
<td>A</td>
<td>V4</td>
</tr>
<tr>
<td>B</td>
<td>V3-V4</td>
<td>A</td>
<td>V6</td>
</tr>
<tr>
<td>B</td>
<td>V5</td>
<td>A</td>
<td>&gt;V7</td>
</tr>
<tr>
<td>C</td>
<td>V1</td>
<td>A</td>
<td>V5</td>
</tr>
<tr>
<td>C</td>
<td>V2-V3</td>
<td>A</td>
<td>V6</td>
</tr>
</tbody>
</table>

To reduce the time-consuming of dependency graph construction, the queried metadata of third-party libraries via PyPi is stored locally as a dependency knowledge database. Each item in the database is represented by a 2-tuple \( < p, v > \). \( p \) is the name of a third-party library. \( v \) is a
Dependency Declaration

Issue Detection and Fixing

Optimized DFS for full-connected path

library knowledge database

dependency declaration file

Fig. 1. The workflow of EasyPip.

Fig. 2. An example of the dependency graph.

2-tuple represented as < vid, td >, where vid is a version of p, and td is a set of third-party libraries with specific versions that p.vid depends on. Before querying PyPi, EasyPip queries the database. If the database contains the information of the third-party library, EasyPip will not query PyPi. Considering that in the actual development, for a developer, there may be many common third-party libraries used in projects. Therefore, with EasyPip used by the developer more times, the local knowledge database can reduce time costs. EasyPip will update the information in the local knowledge database periodically.

Dependency Declaration Issue Detection

Based on the constructed dependency graph, EasyPip detects dependency declaration issues by searching a fully-connected path that satisfies all denoted version constraints in the graph. Note that the interpreter and each library can only have one node on the fully-connected path. The fully-connected path represents the feasible solution satisfying all declared constraints of the dependency declaration file. Therefore, if there is no fully-connected path that can satisfy all denoted version constraints in the dependency graph, EasyPip will report a dependency declaration issue and the conflicting dependencies that cannot find compatible versions within declared version constraints.

To determine the search order that can improve search efficiency and align with the PIP installation policy, EasyPip applies topological sorting to the dependency graph. To do this, the nodes of the same library are regarded as a whole in the topological sorting. For two libraries i and j, if there exists any node of i that connects to the node of j by a dep_edge, the in-degree of library j is added 1. For example, in the Figure 2, the in-degree of library A is 2. EasyPip calculates the degree of each library. The higher the degree of the library, the more constraints from other libraries (we call them pre-constraints) on the library. For the library, EasyPip should search for compatible versions after its pre-constraints are satisfied, which can reduce the number of backward searches. Therefore, EasyPip applies topological sorting to the dependency graph in ascending order of libraries’ in-degree. In this example, the topological order is B, C, and A.

Based on the sorted graph, EasyPip searches for the fully-connected path that satisfies all denoted version constraints. It travels from the starting library or interpreter to the tail library. For the nodes of the same library or interpreter, EasyPip searches from the node with a newer version to an older version within the denoted node range.

To reduce the search space, we introduce the Equivalent nodes, which refers to different nodes of the same library or interpreter that have the same dependency relations with the nodes of other third-party libraries. Taking an example in Figure 2 for library B, B4 and B3 are equivalent nodes. For one node, if there is no fully-connected path containing the node, there is also no fully-connected path containing its equivalent nodes. Therefore, for each library, instead of traversing all nodes, after finishing the search of one node, EasyPip will skip its equivalent nodes.

Dependency Declaration Issue Fixing

When a dependency declaration issue and conflicting dependencies are reported, EasyPip will fix it by a greedy search of nodes in conflicting libraries. Considering the developers tend to make smaller updates to minimize the impact of any breaking changes, the metric function is the distance from...
the original declared version range. To fix the issue with the least modification, for the conflicting libraries, EasyPip will find the compatible nodes closest to the denoted version range. Specifically, EasyPip traverses the nodes of the conflicting library from the upper boundary of denoted node range to the node with the latest version. If EasyPip fails to find the fully-connected path, it will traverse the nodes from the lower boundary of denoted node range to the node with the oldest version. The first fully-connected path found by EasyPip is the closest solution to the declared constraints in the original dependency declaration file, and EasyPip will recommend it as the fixing solution.

Taking Figure 2 as an example, as the dependency declaration requires $B == 5, C$, there is no fully-connected path that satisfies the declared version constraints of $B$. EasyPip fixes the issue by greedy search from $B4$ to $B1$ to find a fully-connected path ($B3$ will be skipped as it is an equivalent node of $B4$). $B4$ is the first node of library $B$ that can constitute a fully-connected with $A$ and $C$, and its version is closest to the declared version range in the original dependency declaration file. EasyPip will recommend $B4, C3, A6$ as the fixing solution.

IV. EVALUATION

To demonstrate the effectiveness and efficiency of EasyPip, we evaluate it on the dataset of dependency declaration files with dependency declaration issues. The dataset contains 67 dependency declaration files, which are from the issues of Pattern A and Pattern B in our empirical study. We answer the following questions:

**RQ1:** How effective is EasyPip in detecting and fixing dependency declaration issues?

**RQ2:** How effective and efficient is EasyPip to detect and fix dependency declaration issues compared to other state-of-art techniques? Whether EasyPip can generate feasible fixing solutions with less modification to original dependency declaration files?

A. Experiment Design

For RQ1, we run EasyPip to detect and fix dependency declaration issues in our dataset.

For RQ2, we evaluate EasyPip in comparison to two recently state-of-art techniques for detecting and fixing dependency declaration problems, SMARTPIP [4] and PyEgo [2]. We select the baselines following the criteria: (1) the tools are available, (2) they should be align with the goals of our research. SMARTPIP [4] is an open-source technique to search for the dependencies’ compatible versions within declared version constraints, which translates concerned libraries’ version constraints collected in a pre-built knowledge base into SMT expressions for dependency resolution, resulting in superior performance compared to PIP no matter pre or post V20.3. If SMARTPIP fails to find compatible versions of dependencies within all declared version constraints, it will report a conflicting issue. We run EasyPip and SMARTPIP to detect dependency declaration issues in the same dataset, and compare their effectiveness and efficiency from the following aspects:

1) How many dependency declaration issues are detected?
2) How long does it take to detect the issue?

While SmartPip cannot provide fixing solutions for dependency declaration issues, to compare the performance of fixing dependency declaration issues, we select PyEgo, a tool to generate feasible solutions for versions of dependencies used in projects according to code snippets. We run EasyPip and PyEgo to fix the same dependency declaration issues, and compare them from the following aspects:

1) How many dependency declaration issues can be fixed successfully?
2) How many modifications are made to the original dependency declaration files?

B. Answer to RQ1

In the 67 dependency declaration files with dependency declaration issues, EasyPip successfully detects 61 of them and reports the root causes. It takes EasyPip 7.5 seconds on average to detect the issue in one dependency declaration file. For the detected dependency declaration issues, EasyPip generates 44 feasible fixing solutions.

We reviewed the 6 dependency declaration files that EasyPip failed to detect issues and found that they were due to some dependencies whose requires_dist on PyPI are missing. To verify our analysis, we manually obtain the corresponding information of them and add it to the library knowledge database. Then EasyPip can successfully detect and report the correct root cause of the 6 dependency declaration issues.

As for the fixing solutions generated by EasyPip, we need to verify their correctness. To do this, for each fixing solution, we installed and built them in the corresponding projects, to check whether it can be successfully installed and built without errors. The 44 fixing files generated by EasyPip are successfully installed and built. For the 17 dependency declaration issues that EasyPip failed to generate fixing solutions, we reviewed the running log of EasyPip and found that they were due to “timed out”. We manually reproduce the fixing of these issues. In the process of searching, some libraries have too many versions which results in a too long time of searching.
C. Answer to RQ2

Compare to SMARTPIP: In 67 dependency declaration issues, SMARTPIP detects 53 of them but does not report the root cause. It takes 25 seconds on average to detect the dependency declaration issue in one file. We reviewed and analyzed the 14 issues that SMARTPIP failed to detect and found that they were due to incompatible versions of Python interpreters. From the comparison results, it can be seen that EasyPip detects dependency declaration issues more efficiently than SMARTPIP. Furthermore, compared with it, EasyPip has the ability to locate the root causes of dependency declaration issues and detect the issues caused by incompatible versions of Python interpreters.

Compare to PyEgo: PyEgo fixes 21 out of 67 dependency declaration issues, and the 21 fixing dependency declaration files are successfully installed and built in corresponding projects. Compared with PyEgo, EasyPip generates more feasible fixing solutions for the same dataset of dependency declaration issues. EasyPip also fixes all 21 issues. To fairly compare EasyPip and PyEgo, we evaluate both tools on the 21 common issues.

To compare the modification to the original dependency declaration file, we define three indicators:

1. **The number of dependency changes**: measures the change in the number of dependencies from the original to the fixing file. For example, if there are two dependencies in the original file, and the number of dependencies in the corresponding fixing solution is three. The number of dependency changes is 1.

2. **The number of dependency version changes**: counts the number of dependencies with version constraints that differ between the original and fixing files. For example, if there is a dependency $d$ that the version constraint of $d$ in the fixing file is different from its version constraint in the original file, the number of dependency version changes is added 1.

3. **The version distance**: quantifies the degree of version changes from the original to the fixing file. For example, the original dependency declaration file is "$A==1, I <=C <=3$" and the fixing file is "$A=2, C=4$", the version distance between the original file and fixing file is 2 (the version distance of $A$ is 1, and the version distance of $C$ is 1).

By comparing the three indicators above of EasyPip and PyEgo on the same dependency declaration files, from Table II, we can conclude that the feasible fixing solutions generated by EasyPip make less modification to the original dependency declaration files.

### Table II. Comparison result with PyEgo

<table>
<thead>
<tr>
<th></th>
<th>PyEgo</th>
<th>EasyPip</th>
<th>PyEgo</th>
<th>EasyPip</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of dependency changes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>avg</td>
<td>max</td>
<td>min</td>
</tr>
<tr>
<td>PyEgo</td>
<td>0</td>
<td>+1.10</td>
<td>+7</td>
<td>0</td>
</tr>
<tr>
<td>EasyPip</td>
<td>-3</td>
<td>-19.2</td>
<td>-55</td>
<td>0</td>
</tr>
<tr>
<td>The number of dependency version changes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PyEgo</td>
<td>EasyPip</td>
<td>PyEgo</td>
<td>EasyPip</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2.90</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>The version distance</td>
<td></td>
<td></td>
<td></td>
<td>PyEgo</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>6.54</td>
<td>2</td>
<td>0.04</td>
</tr>
</tbody>
</table>

V. Threats to Validity

Similar to other bug-related studies [3], [9], the collected data and researchers involved, and keyword search can include irrelevant issues. And manual analysis can introduce bias. To reduce these threats, the researchers carefully reviewed the relevant information and discussed the issues until reaching an agreement. Another threat comes from the collected knowledge. The collected dependencies relationship between third-party libraries and the system libraries may introduce false positives. To alleviate this, we acquire the knowledge following the practice of [2], [10]. And our dataset and analysis results are publicly available. It can help other researchers for further analysis and validate our study results.

VI. RELATED WORK

Dependency Inference. Techniques like DockerizeMe [11], V2 [12] PyCRE [13], and PyEgo [2] can build an environment specification for the code snippet as a Dockerfile. PyEgo conduct static analysis and infer dependencies by solving constraints with Z3. The dependencies they infer must be in their knowledge database, so there is potential conflict among the dependency the developer want but are not listed in the Dockerfile. Different from them, EasyPip only takes the dependency declaration file into consideration for the original dependency declaration file is most close to the environment configuration that the developer expects.

Dependency conflict Dependency conflict (DC) issues hinder the reusability of open-source projects. Researchers have developed various approaches to detect and analyze DC issues [14]–[17]. Watchman [3] characterize DC issues caused by PIP’s installation rule in popular Python projects. Besides them, SMARTPIP is the closest to our work. However, it aims to improve the dependency constraints-solving strategy of PIP. It cannot provide useful information to help locate the root cause of DC issues. And they don’t take the Python interpreter into consideration during dependency resolution. PyDFix [18] concentrated on detecting and fixing unreproducibility in Python builds caused by third-party library errors. They take the old and current build logs as input and try to fix the errors by iteratively trying the other version of the same library. They indicate that it’s efficient to fix dependency declaration issues by adjusting the dependency constraints. To the best
of our knowledge, there is no previous work focusing both on detecting and fixing issues at the same time in the dependency declaration files in the Python world.

VII. Conclusion

In this paper, we design and implement a technique, EasyPip, to automatically detect and fix dependency declaration issues. We evaluate EasyPip with existing state-of-art techniques on both the dependency resolution task and dependency declaration issues fixing on the benchmark and collected dependency declaration issues in real-world Python projects. The results show that EasyPip can efficiently conduct dependency resolution, and detect and fix more dependency declaration issues with fewer modifications to the original dependency declaration files released by developers. In the future, we plan to further improve the detection and fixing capability of EasyPip on more patterns of dependency declaration issues.

VIII. ACKNOWLEDGMENT

We thank Keqin Xu, Zhiyang Zhou and Zhaolin Shi for their contributions. This work was partially supported by National Key R&D Program of China Grant (2017YFA0700603), National Natural Science Foundation of China (61972386).

References

CAPS: An Efficient Whole-Program Critical Paths Search Framework for Large-Scale Software

Peiyang Li, Zixin Liu, Yuening Su, Hao Wang, Bo Jiang*
State Key Laboratory of Software Development Environment
School of Computer Science and Engineering
Beihang University, China
{lipeiyang, liuzixin, Su_Yuening, wangritian, jiangbo}@buaa.edu.cn

Abstract—Tracking the flow of external inputs in a program with taint-analysis techniques can help developers better identify potential security vulnerabilities in the software. However, directly using the static taint analysis provided by Clang Static Analyzer is inefficient for large-scale software due to the huge but redundant ExplodedGraph generated. Therefore, we propose an efficient Whole-Program Critical Paths Search (CAPS) framework. It first performs a set of optimizations to reduce the ExplodedGraph of each function. Then, it constructs a global exploded graph by inserting call edges among the reduced ExplodedGraphs for each function within the Neo4j graph database. Finally, it proposes loop removal and graph segmentation to optimize the search process for critical paths on the global exploded graph. Our experiments on 3 large-scale software show that CAPS can significantly improve the efficiency of critical path search for large-scale software.

Keywords-component; critical path search; static analysis; Clang Static Analyzer; ExplodedGraph; taint analysis

I. INTRODUCTION

For modern software systems, external input data has the potential to trigger its underlying vulnerabilities [1]. Many taint analysis techniques [2][3][4] have been proposed to identify these vulnerabilities or to detect privacy leaks. These techniques typically start by obtaining the control flow graph and data flow graph of the program. Then, external inputs (e.g., input parameters) or other untrusted data are marked as sources on the data flow graph. A taint analysis is performed on the graph to track the impact of the sources within the program until the sinks are reached. A typical taint analysis requires the sinks to be defined and identified in advance. Taint tracking cannot be performed without the knowledge of the sinks. However, defining and identifying sinks for large-scale complex software can be difficult. Furthermore, in our work, we are only interested in identifying all the paths reachable from a source to guide manual security check or fuzzing. Therefore, our goal in this work is to perform whole-program critical paths search for a large software. In this work, the whole-program critical paths are defined as all the paths reachable from each public interface function through static taint analysis by marking each of its parameter as tainted.

Many program static analysis techniques have been proposed for program path analysis, such as Clang Static Analyzer (CSA)[5], SVF[12], Phasar [13], etc. These techniques typically use symbolic execution or data flow analysis methods to explore executable paths in a program. Specifically, CSA is a source code analysis tool that represents all input values (e.g., function parameters) as symbolic values and performs path-sensitive code analysis using symbolic execution techniques. Given the entry function of a software, it explores all possible execution paths in the whole program and calculate the symbolic values of expressions in the program, such that all expressions related to input values are represented as a function of the input symbolic values. The set of explored paths is represented using an ExplodedGraph. However, performing taint analysis by feeding the entry function to Clang Static Analyzer cannot track all the critical paths starting from each public interface function. Furthermore, if we perform inter-procedural taint analysis on each public interface function, many functions will be repeatedly analyzed, leading to explosive growth in analysis time.

To address the above problem, we adopt CSA to perform intra-procedural analysis on all functions to obtain their respective ExplodedGraphs. Next, we extract the entities and entity relationships from each ExplodedGraph and add the call relationships between functions. Then, we import these entities and relationships into the Neo4j [6] graph database to build the global exploded graph for the software. Finally, we perform search on the global exploded graph within the graph database to find all critical paths for an interface function.

However, we found the above approach to find critical paths within the global exploded graph is inefficient due to two reasons. First, the ExplodedGraph contains a large number of redundant nodes, which makes the search space too large to search efficiently. To solve this problem, we propose a technique to reduce the scale of the ExplodedGraph by merging and deleting nodes that are irrelevant to the tainted function parameters. Our experiment shows the technique can reduce the scale of the original ExplodedGraph by approximately 80%. Second, when searching for the critical path on the global exploded graph, we found that there are some loops and repeated paths, which further reduce the efficiency of path search. To solve this problem, we propose an optimization algorithm for path search that can significantly improve its efficiency. Based on the above proposals, we build a whole-program Critical Paths Search (CAPS) framework for efficient critical path search of large-scale software.

The contributions of this paper are as follows:

- We build a custom Checker based on CSA, called ExplodedGraphEmitChecker. The Checker generates a
reduced ExplodedGraph for each functional unit by merging and removing nodes that are not related to the critical path in the original ExplodedGraph.

- We extract entities and their relationships from the reduced ExplodedGraph of each function and add function call relationships, and then use the Neo4j graph database to build the global exploded graph for large software.
- We propose an optimized whole-program critical paths search method based on the depth-first search algorithm. This method reduces the search space through loop deletion and graph segmentation, thereby improving the efficiency for critical path search.
- We have systematically evaluated our framework on 3 large GNU software. The experimental results show that it can significantly improve the search efficiency of critical paths on the global exploded graph.

The remaining sections are organized as follows. In Section II we detail our proposed whole-program critical paths search (CAPS) framework for large software. In Section III we conduct experiments with CAPS on 3 large GNU software and discuss the experimental results. Sections IV and V present related work and conclusions.

II. WHOLE-PROGRAM CRITICAL PATHS SEARCH

In this section, we present the Whole-Program Critical Paths Search (CAPS) framework as shown in Fig. 1. The framework is mainly divided into three modules: ExplodedGraph scale reduction, global exploded graph generation, and path search optimization. Next we will describe each module in detail.

A. ExplodedGraph Scale Reduction

The scale of the original ExplodedGraph is large. Because CSA’s symbolic execution engine generate multiple ExplodedNodes containing the program state and program points for each analyzed statement at the corresponding program location, which can easily cause the rapid growth of the number of nodes. By analyzing these nodes, it can be discovered that there are three types of ExplodedNodes in the original ExplodedGraph that are irrelevant to the critical path: a) **Synonymous nodes.** These nodes are adjacent in location and have the same state information. b) **Channel nodes.** These nodes have a unique predecessor node and successor node, and do not involve branching and merging of paths. c) **Redundant nodes.** These nodes are not related with program input values and do not exist on the critical path. In this module, we generate a scale-reduced ExplodedGraph for each functional unit by processing the above three types of nodes.

_Merging Synonymous Node._ If there is a subgraph $G_s$ in the original ExplodedGraph $G$ where all nodes in $G_s$ have exactly the same expression information and program state information, then all nodes in $G_s$ are merged into a single node $n_s$. The merged node $n_s$ inherits all the previous predecessor and successor relationships between internal nodes and external nodes in $G_s$.

_Merging Channel Node._ If there is a subgraph $G_c$ in the original ExplodedGraph $G$ where all nodes are channel nodes (i.e., each node only has one predecessor and one successor), then all nodes in $G_c$ are merged into a single node $n_c$. The merged node $n_c$ contains all the information of all nodes in $G_c$ and also retains all the previous predecessor and successor relationships between internal nodes and external nodes in $G_c$.

_Deleting Redundant Node._ If there is a node $n_r$ in the ExplodedGraph, and the ProgramState of $n_r$ does not contain any symbolic information about function parameters, then $n_r$ will be deleted, and the original predecessor node and successor node of $n_r$ will be adjacent to each other.

Based on the above method, we reduced the number of nodes in the original ExplodedGraph, thereby reducing the scale of the ExplodedGraph and improving the efficiency of the subsequent critical path search. It should be noted that the above method is proposed under the premise of satisfying the following two principles:

- **Graph isomorphism.** When all nodes in a subgraph $G'$ are merged into a single node $n$, if an external node of the $G'$ is a predecessor node of a node in $G'$, then it is also a predecessor node of node $n$; if an external node of the $G'$ is a successor node of a node in $G'$, then it is also a successor node of node $n$.

- **Information consistency.** If the state information and function call information in the node exists in the original ExplodedGraph, it still exists in the reduced ExplodedGraph.

Graph isomorphism ensures the synonymity of paths in the reduced ExplodedGraph. That is to say, any path passing through any node in subgraph $G'$ in the original ExplodedGraph

![Figure 1. The Whole-Program Critical Paths Search (CAPS) Framework.](image-url)

170
will also pass through the merged node in the reduced ExplodedGraph. Information consistency ensures that any path search strategy on the original ExplodedGraph and the reduced ExplodedGraph before and after reduction will yield consistent results.

B. Global Exploded Graph Generation

We use the graph database Neo4j to build a global exploded graph for large-scale software. In this graph, each node represents an entity in the ExplodedGraph (such as a function, ExplodedNode, expression, or function call point), and each entity has multiple attribute information as shown in TABLE I. Edges represent the relationships between these entities. For example, since an ExplodedGraph corresponding to a function contains multiple ExplodedNodes, we use INCLUDE to represent the relationship between the function and the ExplodedNode, as shown in TABLE II. We extract entities and relationships between them by parsing the reduced ExplodedGraph corresponding to each functional unit. However, there are many functions in a large software, meaning that there are multiple ExplodedGraphs. In order to build the global exploded graph, we merge the parsing results corresponding to all the ExplodedGraphs and import them into Neo4j in batch. In the global exploded graph, if there is a calling relationship between functions, we will connect the corresponding reduced ExplodedGraphs of the functions using the FunCall relationship in TABLE II.

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Entity Attribute Set</th>
<th>Actual Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFG</td>
<td>id, line, moduleName, parameters, szFun, szOrg</td>
<td>Function</td>
</tr>
<tr>
<td>Node</td>
<td>id, line, moduleName, szFun, szpretty</td>
<td>Simplified ExplodedNode</td>
</tr>
<tr>
<td>Exp</td>
<td>id, szExpName, szExpValue</td>
<td>Expression</td>
</tr>
<tr>
<td>FunCall</td>
<td>id, moduleName, szFileOfFuncFirstDecl, szFun, szPrototype, szRareStmt</td>
<td>Function call point</td>
</tr>
</tbody>
</table>

In TABLE I., line represents the actual location of the entity in the source code; moduleName represents the project filename that the entity belongs to; szFun represents the function that the entity belongs to and its parameters; szOrg represents the absolute path of the project file that the entity belongs to; szpretty represents the source code statement; szExpName represents the source expression; szExpValue represents the symbol value corresponding to the source expression; szFileOfFuncFirstDecl represents the file address where the called function is located; szPrototype represents the name and parameters of the called function; szRareStmt represents the function call statement. In TABLE II, ENTRY represents the relationship between a function and its entry node, where each function has only one entry; INCLUDE represents the relationship between a function and its ordinary nodes; NEXT represents the sequential relationship between nodes; EXP represents the relationship between a node and multiple expressions within the node; CALL represents the relationship between a node and its function call points; CALLEE represents the relationship between a function call point and the called function.

C. Path Search Optimization

The complexity of the original ExplodedGraph structure greatly affects the efficiency of path search on the global exploded graph. Because CSA caches and reuses some of the same ExplodedNodes in order to improve analysis performance and avoid repeated calculations during the process of generating the ExplodedGraph of the functional unit. However, this caching and reuse mechanism may lead to a reverse path from the current node to the previous node, forming a loop, which indirectly leads to an infinite loop of the path search algorithm. In addition, the number of paths of ExplodedGraph corresponding to complex functions is usually exponential, which greatly affects the search efficiency. Therefore, in order to improve the search efficiency of the critical path on the global exploded graph, we propose the CAPS framework. CAPS first optimizes the intra-function path search method based on the naive DFS (Depth First Search) algorithm. The optimizations include loop removal and graph segmentation. After the search for intra-function paths is completed, CAPS combines multiple intra-function paths to generate a complete cross-function path based on the global exploded graph, achieving whole-program tracing of the parameters.

The task of searching intra-function paths can be described as $IntraPathSet = searchIntraPath (Func, SelectParamSet)$. Its inputs are a function ($Func$) and a set of parameter symbol values ($SelectParamSet$), and its output is a set of parameter-related intra-function paths. An intra-function path refers to a path that satisfies the following conditions:

- The path is an ordered list of nodes in the ExplodedGraph of a function ($Func$).
- The starting node of the path is the unique entry node of the ExplodedGraph.
- The ending node of a path is the target node in the ExplodedGraph, i.e., the node that contains one of the parameters in $SelectParamSet$.

1) Graph Optimization

Before searching intra-function paths on ExplodedGraph, CAPS sequentially performs the following optimization processes: loop deletion and graph segmentation.

a) Loop Deletion: The loop deletion transforms the ExplodedGraph into a Directed Acyclic Graph (DAG) by deleting some edges from the original ExplodedGraph. Specifically, the CAPS first performs a DFS traversal on the original ExplodedGraph starting from the entry node, to define the priority of all nodes. The later a node is traversed by the DFS, the higher its priority is. Then CAPS deletes all edges that
satisfy the following condition (i.e., loop edges): \( \text{Priority}(st) \leq \text{Priority}(en) \), where \( st \) and \( en \) are respectively the starting and ending nodes of the edge.

b) **Graph Segmentation:** Graph segmentation is another method to solve the problem of exponential intra-function path number. It makes use of redundant information among intra-function paths and changes the path’s data representation. The purpose of graph segmentation is to divide the nodes of a DAG graph into two sets \( S1 \) and \( S2 \), which are close in size (Fig. 2). Graph segmentation satisfies the following principles:

- Any intra-function path ending at \( S1 \) node (a node in set \( S1 \)) contains no \( S2 \) node.
- Any intra-function path ending at \( S2 \) node contains a special \( S1 \) node called barrierNode (Barrier Node), which can divide this path into two parts: the former part is a path contains no \( S2 \) node (An intra-function path ending at barrierNode), the latter part is a path contains no \( S1 \) node.

In this way, all intra-function paths that have the same barrierNode can be jointly represented as \(<\text{FormerPathSet}> \otimes <\text{LatterPathSet}>\). FormerPathSet is a set of several intra-function paths ending at barrierNode. LatterPathSet is a set of several paths contain no \( S1 \) node. The symbol \( \otimes \) represents the Cartesian product of two sets. This path representation can save storage space, and limits the DFS path search range to set \( S1 \) or \( S2 \) (instead of the entire ExplodedGraph). As shown in Fig. 2.

![Figure 2](image_url)

Figure 2. (a) The node set \( S1 \), \( S2 \) and barrier node in graph segmentation. (b) All the paths represented by symbol \( \otimes \) from node 1 to node 8.

2) **Search of Intra-function and Cross-function Path**

The task of intra-function path search can be described as \( \text{IntraPathSet} = \text{searchIntraPath}(\text{Func}, \text{SelectParamSet}) \). Because of the existence of graph segmentation algorithm, the elements of \( \text{IntraPathSet} \) have the form \( \text{innerpath} = <\text{formerpathset}> \otimes <\text{latterpathset}> \). If the ending node of an intra-function path has function call, then the path can be extended to other functions. In the process of cross-function path search, the procedure of \( \text{searchIntraPath}(<\text{Func}, \text{SelectParamSet}) \) is called continuously, and several intra-function paths are combined into a complete cross-function path.

**D. Limitation**

However, we found that the critical path search method, which searches based on the symbol value of function parameters, is limited by the analysis ability of the CSA itself, and sometimes the complete path cannot be tracked. The specific reason is that CSA is currently unable to accurately infer the symbol values of some expressions (e.g., floating-point variables, complex type variables, function return values, etc.), which causes the interruption of the propagation of parameter symbol values.

III. EXPERIMENT AND EVALUATION

In this section, we conduct experiments on 3 large-scale open-source GNU software and evaluate our CAPS.

A. Reasearch Questions

RQ1: Can CAPS effectively reduce the scale of the ExplodedGraph?

RQ2: Can CAPS effectively improve the efficiency of critical path search for large-scale software?

B. Experiment Design

1) **Subject Programs**

We selected three real large-scale GNU software to evaluate CAPS, as shown in TABLE III. Tar [7] is a widely used archiving and packaging software on UNIX and UNIX-like systems, which can combine multiple files into a single file. In addition, Tar also has other file operation functions, such as extraction, storage, etc. Mailutils [8] is a protocol-independent framework for email processing. It provides a set of libraries for doing almost any mail-related task on any existing mailbox format, using a consistent format-independent API. M4 [9] is a macro processor in the sense that it copies its input to the output expanding macros as it goes. Besides just doing macro expansion, M4 has built-in functions for including named files, running UNIX commands, doing integer arithmetic, manipulating text in various ways, recursion etc.

TABLE III. PROJECTS USED IN OUR EXPERIMENT

<table>
<thead>
<tr>
<th>Subject</th>
<th>Version</th>
<th>Loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tar</td>
<td>1.33</td>
<td>102k</td>
</tr>
<tr>
<td>Mailutils</td>
<td>3.13</td>
<td>209k</td>
</tr>
<tr>
<td>M4</td>
<td>1.4.19</td>
<td>142k</td>
</tr>
</tbody>
</table>

2) **Experiment Setup**

We performed our experiments on a desktop with Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz and 32GB of memory. The operating system is Ubuntu 20.04 LTS. The version number of Clang used to generate the ExplodedGraph for the functional unit is 15.0.0. The version number of the graph database Neo4j used to build the global exploded graph for the software is 4.3.12.

3) **Experiment Procedure**

For each open-source software, we first use CAPS to generate a reduced global exploded graph, and then compare it with the unreduced global exploded graph in terms of node and relationship counts to verify the effectiveness of CAPS in achieving ExplodedGraph scale reduction. Next, we use the naive depth-first search algorithm as a baseline to verify the performance advantages of the path search optimization method in CAPS on the reduced global exploded graph.

We define metrics Path Search Rate (PSR) and Weighted Rate Ratio (WRR) to evaluate the search efficiency of search algorithms in critical path search. PSR is the ratio of the number of critical paths found to the search time when searching for the critical path of a parameter of a function in the global exploded
Because it is impossible to calculate the PSR of all function parameters within a limited amount of time in a large software with numerous functions and parameters. Therefore, in order to effectively evaluate the performance of search algorithms in the global exploded graph, we first randomly select 200 functions parameters from the software for critical path search, and choose the parameters with path counts in the top 25% as the experimental validation set for calculating their PSR metrics. We chose parameters that involve more critical path numbers as our validation set because functions with fewer critical paths have very small differences in PSR between CAPS and DFS, which can be almost negligible. Then, we compare the performance of CAPS and naive DFS algorithms according to (1), where PSR$_{CAPS}(i)$ and PSR$_{DFS}(i)$ respectively represent the search rate of these two methods when performing critical path search on the $i$th function parameters.

$$\text{Weighted Rate Ratio (WRR)} = \frac{\sum_{i=1}^{n} PSR_{CAPS}(i)}{PSR_{DFS}(i)}$$

(1)

C. Result and Analysis

In this section, we present our experimental results and analyze the research questions proposed in III(A).

1) Answering RQ1

TABLE IV. presents the comparison results of the number of nodes and edges in the global exploded graph of 3 GNU software before and after reduction. The “original” and “reduced” respectively represent the global exploded graph before and after reduction. The “reduction rate” shows the percentage decrease in the number of nodes or edges in the reduced graph compared to the original one. According to TABLE IV., we can see that CAPS has the best performance in reducing the global exploded graph of M4, and its reduction rate of node and edge counts is 85.14% and 83.47%. Especially the reduction rate of the number of edges is 6.45% and 8.61% higher than that of Tar and Mailutils, respectively. Regarding the reduction of nodes, CAPS shows comparable performance with Tar and Mailutils, achieving reduction of 80.58% and 80.17%. Overall, the average reduction rate of nodes and edges is 81.96% and 78.45% respectively, which indicates the effectiveness of CAPS in ExplodedGraph scale reduction.

TABLE IV. COMPARISON OF THE NUMBER OF NODES AND EDGES BEFORE AND AFTER GLOBAL EXPLODED GRAPH REDUCTION

<table>
<thead>
<tr>
<th>Project</th>
<th>Node</th>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tar</td>
<td>original 20,026,774</td>
<td>138,976,378</td>
</tr>
<tr>
<td></td>
<td>reduced 3,889,400</td>
<td>31,938,078</td>
</tr>
<tr>
<td></td>
<td>reduction rate 80.58%</td>
<td>77.02%</td>
</tr>
<tr>
<td>Mailutils</td>
<td>original 25,972,727</td>
<td>186,716,558</td>
</tr>
<tr>
<td></td>
<td>reduced 5,149,298</td>
<td>46,940,666</td>
</tr>
<tr>
<td></td>
<td>reduction rate 80.17%</td>
<td>74.86%</td>
</tr>
<tr>
<td>M4</td>
<td>original 12,027,333</td>
<td>86,391,874</td>
</tr>
<tr>
<td></td>
<td>reduced 1,787,609</td>
<td>14,360,459</td>
</tr>
<tr>
<td></td>
<td>reduction rate 85.14%</td>
<td>83.47%</td>
</tr>
</tbody>
</table>

Furthermore, from the reduction rate of the number of nodes, we can further analyze that most of the nodes in the original ExplodedGraph are irrelevant to function parameters. Therefore, by optimizing and removing these nodes, we can greatly improve the efficiency of subsequent critical path search.

2) Answering RQ2

TABLE V. shows the weighted rate ratio (WRR) of the 3 GNU software after critical path search using CAPS and naive DFS algorithms at five different search time of 30s, 60s, 90s, 120s and 150s. Larger WRR indicates that CAPS is more efficient than naive DFS in critical path search. According to TABLE V., we can see that for Mailutils, the average WRR is the highest, reaching 41.56. On the other hand, for Tar, the average WRR is lowest, only 7.39. Furthermore, for the same project, the WRR does not increase linearly with the increase in search time. These indicate that the performance advantage of CAPS over the naive DFS algorithm may be affected by the intrinsic structure of the software.

When we observe the WRR under different software and different search time, we can find that their values are all larger than 1, and the minimum value is 1.82, which shows that CAPS outperforms the naive DFS algorithms in critical path search efficiency, regardless of the software and the search time. From the above experimental results, we conclude that the CAPS is capable of effectively improving the search efficiency of the critical path for large-scale software.

TABLE V. COMPARISON OF WRR AT DIFFERENT SEARCH TIME

<table>
<thead>
<tr>
<th>Project</th>
<th>30s</th>
<th>60s</th>
<th>90s</th>
<th>120s</th>
<th>150s</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tar</td>
<td>4.16</td>
<td>1.91</td>
<td>1.82</td>
<td>25.46</td>
<td>3.6</td>
<td>7.39</td>
</tr>
<tr>
<td>Mailutils</td>
<td>40.18</td>
<td>7.3</td>
<td>34.34</td>
<td>37.64</td>
<td>88.35</td>
<td>41.56</td>
</tr>
</tbody>
</table>

IV. RELATED WORK

In this section, we review related works on taint analysis and global exploded graph generation.

A. Taint analysis

Taint analysis techniques [10][11] are commonly used to identify information flows in programs, tracking the movement of sensitive information from a set of sensitive sources to sensitive sinks. Sui et al. [12] have developed a static analysis tool called SVF, which integrates the functionalities of pointer analysis, value flow analysis, and taint analysis. The tool takes LLVM IR (Intermediate Representation) as input and uses analysis modules such as Program Assignment Graph (PAG) and Control Flow Graph (CFG) provided by LLVM to perform whole-program pointer analysis, draw Sparse Value-Flow Graph (SVFG) value flow graph, and ultimately obtain the flow representation of each data element within the program. Phasar [13] is a large-scale C/C++ program static analysis framework developed by Philipp et al. It performs data flow and control flow analysis based on the LLVM-IR of the input program to obtain the Interprocedural Control Flow Graph (ICFG) data flow graph. Then, the framework uses the Interprocedural Finite Distributive Subset (IFDS) algorithm to achieve complete taint analysis. Additionally, Phasar also includes a simple Boomerang pointer analysis tool, enabling it to detect some synonym pointer propagation during program execution and fill in the missing parts of the IFDS algorithm analysis. She et al. [14] proposed a novel end-to-end method to track information flow by using neural network, called Neutaint. It models target program computations that occur between taint sources and sinks, and...
automatically learns information flow by observing a set of different execution traces. Experimental results show that Neutaint can achieve an average accuracy rate of 68%. Zhang et al. [15] developed FastDroid, a tool for detecting sensitive data leaks in Android applications. It first constructs a taint value graph (TVG) by flow-insensitive taint analysis to describe the taint propagation process. Then, potential taint flows are extracted from TVG. Finally, it compares the potential taint flows with the control flow graph to obtain the real taint flows. The results show that FastDroid can improve the analysis efficiency while ensuring high precision and recall. However, compared with CAPS, most of the above methods are only suitable for local program analysis, and cannot perform whole program critical path search and taint analysis.

B. Global exploded Graph generation

The global exploded graph [16] can describe program paths within functions as well as between functions through edges. Building the global exploded graph can help analysis tools more accurately understand program behavior and achieve global static analysis [17]. Gharibi et al. [18] developed a program analysis tool called code2graph that can automatically analyze source code, construct its static call graph, generate all possible execution paths of the system, and calculate their similarities. Abdelaziz et al. [19] designed a toolkit for building code knowledge graphs called GraphGen4Code. GraphGen4Code uses generic techniques to capture code semantics, and key nodes in the graph represent classes, functions, and methods. Edges represent the call relationship between functions. It can serve applications such as program search, code understanding, error detection, and code automation. However, the program graph constructed by the above methods based on the control flow graph and call graph, and its granularity is relatively coarse. In contrast, the global exploded graph constructed by CAPS is composed of program point and program state, and the granularity is smaller, thus providing more accurate analysis.

V. CONCLUSION

Many taint analysis techniques have been proposed to track the flow of external inputs in the program, so as to identify potential security vulnerabilities in the software. However, these techniques require the sinks to be defined and identified in advance, which is difficult for large-scale software. If static taint analysis is performed directly on the original ExplodedGraph generated by Clang Static Analyzer, although the sinks does not need to be identified, there are many nodes in the original ExplodedGraph that are irrelevant to external input, which makes the search of the critical path inefficient. Therefore, we propose an efficient Whole-Program Critical Paths Search (CAPS) framework. The framework first implements ExplodedGraph reduction corresponding to each function through node merging and deleting. Then, it utilizes the entities and relationships existing in the reduced ExplodedGraph of each function and the calling relationship between functions, to construct a global exploded graph for large-scale software within Neo4j graph database. Finally, it optimizes the critical path search process by loop removal and graph segmentation. Our experimental results on 3 large GNU software demonstrate that CAPS can significantly reduce the original ExplodedGraph scale and improve the efficiency of critical path search on the global exploded graph for large-scale software.

For future work, we plan to explore more efficient algorithms and optimization techniques to improve the efficiency of whole-program critical paths search. Furthermore, we will attempt to resolve the breakpoint issue currently encountered during critical path searching to enhance the completeness of the path.

REFERENCES

Detect Stack Overflow Bugs in Rust via Improved Fuzzing Technique

Zhiyong Ren  
School of Computer Science  
Fudan University, Shanghai, China  
20210240038@fudan.edu.cn

Hui Xu  
School of Computer Science  
Fudan University, Shanghai, China  
xuh@fudan.edu.cn

Abstract—Stack overflow has been a common memory vulnerability for a long time due to limited stack memory. Deep or infinite recursion serves as the main cause to exhaust the stack memory and crash the program. As a relatively new system programming language, Rust suffers from stack overflow problem inevitably. However, there is no relevant tool to detect those stack overflow bugs in Rust programming language. In this paper, we propose a novel approach using fuzz technique to trigger stack overflow bugs in Rust projects. We first build a call graph on Rust MIR (Middle Intermediate Representation). In the call graph, recursions appear as cycles lying in the SCCs (strong connect components). To find the entry APIs of those SCCs, we leverage Tarjan's algorithm to locate the SCCs and then reversely BFS (Breadth First Search) to search for the APIs. After that, we modify the underlying logic of AFL (American Fuzzing Loop) to trigger stack overflow bugs through fuzzing those dangerous APIs. Specifically, we add a function call time counter to accelerate the fuzzing process. We conduct our experiments on several existing Rust CVEs (Common Vulnerabilities and Exposures) related to stack overflow. Experiments show that our approach can trigger stack overflow bugs in a short time.

Index Terms—fuzz, Rust, recursion, stack overflow

I. INTRODUCTION

Stack overflow is a well-known memory vulnerability for a long time. In software, stack overflow occurs when stack memory is exhausted. Essentially, stack is a linear data structure that follows the principle of LIFO (last in first out). It occupies a limited amount of virtual address space when program runs, e.g., default 8MB on Linux and 1MB on Windows. The size of the stack is determined at the start of the program and depends on many factors, including machine architecture, programming language and the amount of available memory. It grows from top memory address to lower memory address in memory space, which is totally opposite of heap memory allocation. When a program attempts to use more space than is available on the stack, the stack is said to overflow, typically leading to a program crash.

The program embraces a risk of stack overflow when it contains recursion. Unconditional recursive functions, like Figure 1, overflow the stack due to its infinite invocation. Besides, those conditional recursive functions like fibonacci recursive function, whose invocation depends on their context, can also trigger stack overflow errors. Hackers may construct a nested input maliciously according to the control flow of the program and make the functions recurring deeply to crash this program.

Recursion serves as a common programming skill and exists in many projects. Forbid programmers from using recursion seems unfeasible in real-world programming. We program concisely using recursion. Correspondingly, we have to sacrifice the performance of the program due to the cost of calling a function and tolerate the hidden risk of stack overflow. As an emerging programming language that promotes memory-safety features, Rust has attracted many developers in recent years. With no runtime and garbage collector, it empowers performance critical services, runs on embedded devices and easily integrates with other languages. Its rich type system and ownership model guarantee memory-safety and thread-safety features, which enables developers to eliminate many classes of bugs during compile-time. Nowadays, many developers use Rust to build their project due to its memory-safety benefits.

However, just like other programming languages, Rust suffers from stack overflow vulnerability inevitably. Dozens of Rust projects use recursion to implement their logic, e.g., yaml-rust[14], serde[15], ammonia[16]. Eight stack overflow related CVEs are found in Rustsec (The Rust Security Advisory Database). Due to its memory-safety promise, Rust has low tolerance for memory-safety problems. Unfortunately, there is no relevant tool to detect those stack overflow bugs in Rust projects.

In this paper, we detect those stack overflow bugs in Rust crates via an improved fuzz approach. We conduct our experiments on several Rust CVEs and successfully trigger stack overflow bugs in those CVEs within a short time.

We summarize the contributions of this paper as follows:
We leverage Rust type information on MIR to build our call graph and deal with some dynamic features.

- We design an algorithm to find all the entry APIs of SCCs in the call graph for constructing fuzz targets.
- We modify the underlying logic of AFL to accelerate fuzz process of triggering stack overflow bugs in Rust CVEs.

The rest of our paper is organized as follows. Section 2 presents our motivation and related work. After that, we introduce our approach in section 3. Section 4 demonstrates our evaluation experiments and Section 5 introduces limitations and future work. We conclude our paper in section 6.

II. MOTIVATION AND RELATED WORK

A. Motivation

As a static strongly typed language [5], Rust employs a stack to store some of its data. Especially, the Rust compiler puts those data types with fixed size, such as i32, u32, on the stack memory during compile-time, while allocates heap memory for those dynamic data types at run-time. When a function gets called in Rust, some stack memory gets allocated for its local variables, which is called a stack frame. Stack overflow occurs when large amount of frames accumulates through recursive function calls.

Finding a recursive function in Rust crates seems troublesome due to the large amount of functions and complicated function call relations. Unfortunately, we lack a useful tool to automatically detect those unrevealed recursions in Rust crates. To our best knowledge, rustc (the Rust compiler) only provides warnings for those simple unconditional self-recursive functions. Wu[23] develops a clippy lint on HIR to detect those conditional self-recursive function. This lint simply compares the equality of caller def_id and callee def_id. Cargo-call-stack [21] develops a call graph on LLVM IR to analysis stack usage of each function, but it evades recursive functions.

To our knowledge, there is no formal work to test those recursions in Rust crates. Two open issues in the official rust-lang repository (issue 57965[18], 70727[17]) discuss those recursion problems but do not tackle it.

III. APPROACH

In graph theory, a strong connected component denotes a set of vertices and edges in which vertices can reach each other. Basically, one strong connected component consists of serveral simple cycles. It can be decomposed into a certain number of independent cycles. Those cycles serve as recursions in real-world programs and have stack overflow risk. In the call graph, all recursions lie in the SCCs. So we choose to fuzz those SCCs in order to trigger stack overflow bugs. Our approach consists of three parts, including call graph construction, entry APIs searching and fuzzing. This section demonstrates these three parts in detail.

A. Rust Call Graph Construction

Call graph acts as a presentation of relations between functions in program and behaves as a high-level approximation of its run-time. As a normal static analysis approach, call graph presents an overview of the whole program and facilitates inter-procedure analysis.

Rust compiler (rustc) leverages a query system to enable demand-driven compilation. Incremental compilation implemented by rustc accelerates compiling process. During compilation, Rust source code goes through AST (Abstract Syntax Tree), HIR (High-level Intermediate Representation), MIR, LLVM IR [7] forms and finally turns into binary code. Rustc executes different operations on each compiling phase. For example, it does name resolution on AST while performs borrow check on MIR. We can choose...
to build our call graph on these four phases (AST, HIR, MIR and LLVM IR).

Due to the absence of Rust-unique type information, call graph built on AST, HIR or LLVM IR cannot resolve monomorphized features correctly, resulting in imprecise call edge. Therefore, we choose to build our call graph on Rust MIR and leverage type information to make call graph more accurate.

To deal with dynamic features such as dynamic dispatch in Rust, we collect all the types which implement the specific trait. For example, struct A, B, C implement trait BoundTrait. We cannot decide which function to call in compile time due to the absence of type information. So we take all the types (A, B, C) into account to make our call graph sound.

To accelerate the construction process, we cast away functions which is calling into Rust std crate, e.g., println!.

We present the call graph construction process in Figure 2.

B. Entry APIs Searching Algorithm

Cycles lie in strong connected components in call graph. To verify the risk of stack overflow bugs of the cycles, we need to find the entry points of all SCCs. We first leverage Tarjan’s algorithm to find all the SCCs in the call graph. Then we reverse the direction of edges in the call graph.

After that, we remove the edges in SCCs to avoid fruitless search and use BFS (breadth first search) to search for the entry APIs of each SCC. In Figure 3, {a, b, c, d, e, f, g, h, i, j} forms an SCC and {x, s, z} denotes the entry APIs of this SCC. We demonstrate this algorithm in Algorithm 1.
Algorithm 1 Search entry APIs of all SCCs

Input: Adj: adjacency list for a call graph
Output: entryAPIs: the entry API of each SCC

1: SCCs := Tarjan(Adj) /* find SCCs */
2: reverse the edges of the call graph
3: for each scc in SCCs do
4: remove the edges in the scc
5: BFS to find entry APIs
6: end for
7: output those APIs

C. Fuzzing

To trigger stack overflow bug, we fuzz those APIs found by our searching algorithm. We choose AFL to do our fuzz job. To facilitate fuzzing process, we modified the underlying logic of AFL, including instrumentation and seed selection strategy. Specifically, we list the modifications as follow:

- **Instrumentation**: we extra instrument at the entry of each function and assign a unified ID 0 to these instrumentation. For other instrumentation lying in the branch, we generate a random ID from 1 to MAP_SIZE through a built-in function R(x). R(x) means generating a random number from 0 to x. So we set x with MAP_SIZE-1 to generate a random number from 0 to MAP_SIZE-1. Then we plus 1 to generate a number from 1 to MAP_SIZE. The whole expression is R(MAP_SIZE-1)+1. MAP_SIZE denotes the size of the trace_bits. So in trace_bits, trace_bits[0] denotes the times of function calls and trace_bits[1..MAP_SIZE] denotes the trigger times of each branch.

- **Seed Selection**: traditional seed selection strategy uses has_new_bits() function to determine whether the seed covers a new branch after one fuzzing loop. In addition to this, we take the function call times into consideration. That is, if a seed increases the maximum function call times after one fuzz loop, it will be added into the test queue for next mutation. Or it will be selected only if it increases the branch coverage. We use trace_bits[0] to record current function call times and a global variable to record maximum function call times.

Subsequently, AFL will carry out a large number of mutation on the selected seed and check whether it causes crash or finds new path. The main types of mutation includes bitflip, arithmetic, interest, dictionary, havoc, splice and so on. AFL will repeat the fuzzing loop until developers manually stop or timeout.

IV. EXPERIMENTAL EVALUATION

This section presents our experiments on Rust crates. We evaluates our approach for three research questions.

- **RQ1**: How many dangerous APIs are there in our experimental crates?
- **RQ2**: Can our approach trigger stack overflow bugs successfully?
- **RQ3**: How much does our approach speed up the fuzzing process?

A. Experiment Setting

We collect eight Rust stack overflow related CVEs from RUSTSEC and download all related crates from GitHub. Recursion causes all of these CVEs. Users can crash the crates by constructing a nested input maliciously. Developers have located some of those problematic recursive functions in these CVEs and fixed them by limiting the recursion depth or changing the recursion into iterative form. Still, it remains questionable for some CVEs since it is not easy to locate the problematic recursions in dozens of functions with complicated call relations.

Before we test our approach on these crates, we make a new file named rust-toolchain.toml into the root directory of each crate to change the Rust toolchain into right version.

To trigger stack overflow bugs, we choose to fuzz those problematic CVEs. During the fuzzing process, we record the crash times every minutes. In order to speed up the fuzzing process, we use ulimit - s xxKB command to reduce the stack space into 512KB, 1024KB, 2048KB respectively. Moreover, we set the initial input with '{'. At the same time, we compare the unchanged AFL with improved AFL and observe the difference between these two experiments.

We conduct all our experiments on Dell-OptiPlex-7070 with 32GB memory and Intel Core i7-9700T 2.00GHz CPU.
TABLE I
THE NUMBER OF DANGEROUS APIs IN CVE CRATES

<table>
<thead>
<tr>
<th>CVE ID</th>
<th>Crate Name</th>
<th>number of APIs</th>
<th>dangerous</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2019-25001</td>
<td>serde_cbor</td>
<td>27</td>
<td>451</td>
<td></td>
</tr>
<tr>
<td>CVE-2019-15442</td>
<td>markup5ever3</td>
<td>1</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td>CVE-2020-35858</td>
<td>prost_build</td>
<td>1</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>CVE-2018-20994</td>
<td>trust-dns-proto</td>
<td>3</td>
<td>1390</td>
<td></td>
</tr>
<tr>
<td>CVE-2018-20993</td>
<td>yaml-rust</td>
<td>2</td>
<td>241</td>
<td></td>
</tr>
<tr>
<td>CVE-2020-35857</td>
<td>trust-dns-server</td>
<td>3</td>
<td>1390</td>
<td></td>
</tr>
<tr>
<td>RUSTSEC-2018-0005</td>
<td>serde_yaml</td>
<td>21</td>
<td>680</td>
<td></td>
</tr>
<tr>
<td>RUSTSEC-2022-0004</td>
<td>rustc-serialize</td>
<td>3</td>
<td>1504</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II
THE NUMBER OF RECURSIONS IN CVE CRATES

<table>
<thead>
<tr>
<th>CVE ID</th>
<th>Crate Name</th>
<th>number of recursions</th>
<th>self</th>
<th>cross-func</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2019-25001</td>
<td>serde_cbor</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CVE-2019-15442</td>
<td>markup5ever3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CVE-2020-35858</td>
<td>prost_build</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CVE-2018-20994</td>
<td>trust-dns-proto</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CVE-2018-20993</td>
<td>yaml-rust</td>
<td>1</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>CVE-2020-35857</td>
<td>trust-dns-server</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RUSTSEC-2018-0005</td>
<td>serde_yaml</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RUSTSEC-2022-0004</td>
<td>rustc-serialize</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

B. RQ1: How many dangerous APIs are there in our experimental crates?

In real-world Rust programs, Rust developers always define those recursions using private keyword. They often offer an entry API to users at the top layer. This API is always set with public. We need to find these APIs to construct fuzz target.

We run our self-created algorithm to find the dangerous APIs of crate related to Rust stack overflow problem. Those dangerous API is defined as the entry APIs for all SCCs in the function call graph.

Experiment results in Table 1 show that each eight CVE related crate contains at least one dangerous API. In particular, serde_cbor shows the highest number of dangerous APIs, which is up to 27.

Those dangerous APIs lead to the recursions lying in SCCs. The recursions in Rust crate contain not only simple self-recursion but also complicated cross-function recursion. Table 2 demostrates the number of recursions in those CVE related Rust crate. All these recursions have the risk of stack overflow problem.
C. RQ2: Can our approach trigger stack overflow bugs successfully?

We have tested our approach on eight Rust stack overflow related CVEs. These crates contain library crates and binary crates. Figure 5 demonstrates our experiments on each CVE.

Results show that our approach has triggered stack overflow bugs. The blue line represents the experimental results at the stack space of 512KB, while the green line represents 1024KB and the blue line represents 2048KB. At the initial stage, the number of crashes is zero. After roughly eight minutes, the program crashes begin to occur, and the growth rate becomes faster and faster. The reason for the speed may be that the input generated by the mutation can cause the stack overflow bugs already, so each subsequent mutation can also cause stack overflow bugs. Moreover, the larger the stack space is, the longer time it takes to generate a crash.

We have carefully verified the correctness of each input generated by our approach. Specifically, we pass the input into the dangerous API, and it successfully triggers stack overflow bugs in the running process of the program.

D. RQ3: How much does our approach speed up the fuzzing process?

In Figure 5, the red, black and pink lines are the baselines which represent the unimproved AFL in the corresponding stack space. Compared with unmodified AFL, the modified AFL has a significant improvement in fuzzing speed. Take CVE-2018-20993 as an example, at the stack size of 512KB, the first crash occurs at minute 10 under the modified AFL while it occurs at minute 25 under the unmodified AFL. After five minutes, crash times grow into 16 while it only grows into 9 in baseline. At the time of 30 minute, the crash time reaches 312, which is far more than 9 in the baseline.

In other CVEs, it shows the same trend as it is in CVE-2018-20993. We see a clear gap of the experiment result between the improved AFL and baseline under both 512KB, 1024KB and 2048KB.

V. CONCLUSION

In this paper, we propose a novel approach to detect those stack overflow bugs in Rust crates. We first build a call graph on Rust MIR and use our algorithm to locate all entry APIs of SCCs in the call graph. Then we modify the instrumentation and seed selection strategy of AFL to accelerate the fuzz process of triggering stack overflow bugs. We conduct our experiments on Rust stack overflow related CVEs. Experiment results show that our approach successfully triggers stack overflow bugs under the stack size of 512KB, 1024KB, 2048KB. Our approach is the first attempt to tackle stack overflow problem in Rust programming language and has positive effect on further research.

REFERENCES

[19] Tarjan, Robert, Depth-first search and linear graph algorithms, SIAM journal on computing, 146-160, SIAM.
An Efficient Design Smell Detection Approach with Inter-class Relation

Hao Zhu, Yichen Li, Jie Li and Xiaofang Zhang∗
School of Computer Science and Technology
Soochow University, Suzhou, China
{hzhu721,ycli1024,2027407061}@stu.suda.edu.cn and xfzhang@suda.edu.cn

Abstract—Code smell indicates the potential designed problems and quality of source code affecting the software maintenance and readability. Hence, detecting code smells in a timely and effective manner can provide guides for developers in refactoring. Existing methods generally treat these code smells from different granularities equally by merely extracting tokens-based or abstract syntax tree (AST)-based code representation, which does not take the diversity of code smells into account, especially when fewer researches concern design smells. To tackle this challenge, we propose Design Smell Detection through Inter-class Relation, which leverages the corresponding design smells features for code smell detection. More specifically, we employ AST-tokens instead of traditional word-tokens or AST to obtain code syntax information from the deep dimension. Meanwhile, we analyze the common structural feature of design smells and propose the inter-class relation among different class files contained in the same package. Moreover, to verify the effectiveness of our proposed method, we carry out extensive experiments with various settings on our new dataset and the results demonstrate that our method outperforms state-of-the-art methods by up to 31% in terms of F1-measure of all code smells. The code is available at: https://github.com/xzb777/designSmellUML

Index Terms—Code smell, code representation, inter-class relation, deep learning

I. INTRODUCTION

Code smells are indicators of poor coding potentially affecting software quality and maintenance [1]. Therefore, code smell analysis, which is developed to assess and improve software quality by detecting and removing code smells, is of great importance in software design. Software engineering researchers have conducted extensive researches on code smells, including their definition, causes, effects, detection and refactoring [2].

Researchers have devised a variety of methods for detecting code smells, including traditional metric-based methods [3], [4] and heuristic-based methods [5]. As an example, Moha et al. [6] devised DECOR to define code smell detection rules with a specific language. Nevertheless, the majority of these methods suffer from developer’s subjective illustration and threshold set to identify smelly instances. To address this issue, machine learning methods [7], [8] and deep learning methods [9] have recently attracted increasing attention for their efficiency in extracting hidden patterns from large amounts of data for prediction. Guggulothu et al. [10] apply Random-Forest to code smell detection with tradition metrics. Xu et al. [11] adopt the syntax information in AST to optimize the code smell detection from different granularities. Li et al. [12] apply a hybrid model with comprehensive code features to multi-label code smell detection. Though remarkable performance has been achieved, these methods all focus on implementation code smells like Long Method, Missing default, ignoring design smells to some extent. For larger-granularity design smells such as Broken Hierarchy and Insufficient Modularization, they still require careful consideration which usually involve the organization and relation between classes in a software system.

In this paper, we seek to develop a simple and scalable technique to detect design smells. Token-based methods [11] and AST-based methods [13] have had outstanding performance in implementation smells detection with tokens or AST. The rendering of design smells is often accompanied by more complex and longer code snippets, which makes the above methods hard to precisely extract features in practice. A direct idea to solve the problem is to optimize extracted features from the deep dimension and make use of the relation between classes in the same package. In addition, existing public datasets on design smells [12] are split into method/class-level code snippets, losing the relation between classes during data preprocessing.

Considering these limitations, we in this paper propose a novel approach Design Smell Detection via Inter-class Relation (DSIR)—that can efficiently extract features in design smells for detection. Specifically, we first parse the source code into AST and obtain a sequence of node tokens by preorder traversal. Then, we create a class-level semantic graph for each target class based on the UML class diagram. To better extract hidden patterns, we separately apply bidirectional long-short term memory network with attention mechanism (BiLSTM) and relational graph convolution network (R-GCN) and combine the outputs of the two models by weight to obtain the prediction. Besides, we split 500 Java projects on GitHub to build a new dataset with class-relation information and conduct our experiments on it. Experimental results illustrate that our DSIR model outperforms the state-of-the-art methods by up to 31% in terms of F1-measure on the three selected design smells. The major contributions of this paper are summarized as follows:

• We present the appropriate feature extraction for design smells and innovatively apply the class-level semantic
graph based on UML class diagram to obtain inter-class relation for design smell detection.

- We build a new dataset with design smells and preserve class relation for further research at a large granularity.
- We conduct extensive experiments with various settings and baselines. The results demonstrate the effectiveness of the proposed method which improves the F1-measure by up to 31% compared to state-of-the-art methods.

II. RELATED WORK

A. Code Smell

Fowler et al. initially introduced the concept of code smell as structures with technical debt. Code smells serve as indications of deteriorating software quality. Based on granularity, scope, and impact, code smells can be classified into implementation [1], design [14], and architecture smells [15]. Previous research has primarily focused on detecting implementation smells, with excellent results achieved in this regard. However, design smells have a more extensive scope, and identifying and refactoring them may require working with a set of classes.

B. UML Diagram

UML (Unified Modeling Language) is a software engineering modeling language used for describing and designing software systems [16], [17]. Class diagrams, as one of the UML diagrams, are particularly useful for describing the attributes, methods, and relation among classes and interfaces in a system. The relationships in a class diagram, such as dependencies, inheritances, aggregations, and associations, can assist software designers in identifying the structural composition and functionality of a system. However, how to apply UML to code smell detection tasks remains a challenge.

C. Motivation

Design smell is usually expressed as more complex and longer code snippets. Existing token-based methods simply treat code snippets as a natural language to get the structural information. Meanwhile, AST-based methods rely on traversing the AST and analyze the nodes and edges to obtain the syntactic information of the code. However, these approaches are challenging as they require a deep and entire understanding of the AST structure or code tokens which is difficult with long code snippets. Intuitively, using AST tokens instead of code tokens can provide a more expressive representation of the source code and it can help to reduce the feature size of the AST-based approach. Furthermore, We consider that introducing the information from class diagrams can better capture inter-class relation which can help the model learn the hidden patterns of complex class snippets. An example can be seen in Figure 1. This is a class diagram provides the relation between classes in a program, including inheritance and dependencies. Each box in the diagram represents a class in the program, and the name of the class is at the top of the box. The properties and methods of the class are listed in the second and third sections of the box respectively.

![Fig. 1: Example of a class diagram with design smell.](image)

We use the example of an abstract class `ServiceConfiguration` in Figure 1 to illustrate how the class diagram can help us detect design smells. `ServiceConfiguration` is labeled with `Broken Hierarchy` design smell in our dataset. `Broken Hierarchy` refers to the presence of poor inheritance or nesting relation in the code. Then we can see that class `ServiceConfigurationImpl` inherits from `ServiceConfiguration` because they are connected by a line of inheritance, and class `Suppressible` and class `ServiceConfiguration` depend on `ServiceConfigurationImpl`, indicating that `ServiceConfiguration` plays an important role in the class hierarchy and functionality. However, the class `Suppressible` also depends on `ServiceConfigurationImpl`, which breaks the functionality and inheritance of parent and child classes which is consistent with the definition of `Broken Hierarchy`. Therefore, to obtain inter-class relation to detect design smells, we create a class-level semantic graph representation based on class diagram information.

III. METHODOLOGY

A. Problem Formulation

In this section, we introduce our DSIR method to detect design smells. The overview framework is shown in Figure 2. We are aimed to detect design smells including `Broken Hierarchy`, `Insufficient Modularization`, and `Deficient Encapsulation`. To tackle this problem, we divide it into two sub-problems: The first sub-problem involves extracting appropriate features of the source code effectively and we apply the bidirectional long short-term memory network. The second sub-problem involves modeling the relation with relational graph convolutional network between classes based on the class diagrams. Finally, we identify smelly instances based on the fusion model.

B. LSTM Model

We first use `Javalang` to parse class-level code fragments into their corresponding AST. Then, we obtain all the AST node tokens by preorder traversal and use them as inputs to be fed into an LSTM model. The BiLSTM captures both forward and backward dependencies between tokens, while the attention mechanism focuses on the most relevant parts of the input during the encoding process. Specifically, we use global

1https://github.com/c2snet/javalang
attention to extract the source context vector $c_j$ by computing the attention weights $a_{ij}$ of hidden state $h_i$.

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

where $x$ refers to the AST token sequence. The attention mechanism will assign more weight to the hidden state vectors of important tokens.

$$r_{ij} = h_i \ast c_j \quad \quad (2)$$

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

where $r_{ij}$ represents the relevance score between the $i$-th hidden state $h_i$ in the source sequence and the context vector $c_j$ for the $j$-th target token. We then pass $r_{ij}$ through a Sigmoid layer with parameters $W_s$ and $b_s$ to get the output of the model.

C. R-GCN Model

This section presents our approach to detecting design smells using class diagram information to obtain the inter-class relation. We select four common types of relation between classes: dependency, inheritance, association, and implementation, as they are widely used to investigate the metric of inter-class relation in [18]. First, we find the corresponding class diagram for the input class in our dataset, and then we select a set of classes adjacent to it. We then add four common edges to the set of selected classes based on the class diagram. Meanwhile, each class includes its type, name, attributes, and methods in a class diagram. Thus, we extract this information for the selected classes and organize them into an input tuple:

$$\text{input} = (\text{class\_type}, \text{attributes}, \text{methods})$$

$$\text{attributes} = \{\text{attribute}_1, \text{attribute}_2, \ldots, \text{attribute}_n\} \quad (4)$$

$$\text{methods} = \{\text{method}_1, \text{method}_2, \ldots, \text{method}_n\}$$

Here, class\_type represents the type of the class including concrete classes, interfaces and abstract classes. The class’s attributes include different constants or variables, while the class’s methods are input in the form of “method\_name (method\_return\_type)”, where $i$ represents the $i$-th attribute or method information of the class and $n$ represents the number of attributes or methods. We input the tuple into a Transformer-based sentence embedding model to obtain the class semantic embedding and use the embedding vectors corresponding to the classes in the set as the node feature matrix. Finally, we use the Python package PyG to convert these information into a graph, which serves as the input to the R-GCN. Relational Graph Convolutional Networks (R-GCN) is designed to process graph-structured data with complex relationships, allowing us to better capture the inter-class relation. The propagation rule for R-GCN is given by:

$$H_i^{(l+1)} = \sigma \left( \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} H_j^{(l)} \right),$$

where $N_i^r$ denotes the set of neighbors of node $i$ via edge type $r$, $c_{i,r}$ is a normalization constant that scales the contribution of each neighbor according to its degree, and $\sigma(.)$ is an activation function. We can stack multiple R-GCN layers by repeating this propagation rule. Our forward model for R-GCN then takes the form:

$$Z_r = \tilde{A}_r \text{ReLU}(\tilde{A} r X W_r^{(0)} W_r^{(1)}), \quad \forall r \in R \quad (6)$$

$$Z = \sum_{r \in R} Z_r \quad (7)$$

$$y = \text{Sigmoid}(W_s Z + b_s) \quad (8)$$

where $Z_r$ represents the feature vector associated with relation $r$, $\tilde{A}_r$ represents the adjacency matrix of relation $r$ plus a self-connection adjacency matrix, $X$ represents the input feature matrix, $W_r^{(0)}$ and $W_r^{(1)}$ represent the weight matrices of the input and output layers of relation $r$, respectively. ReLU is the rectified linear unit function, $Z$ is the vector obtained by summing up all the relation feature vectors, $W_s$ and $b_s$ are the weight and bias of the output layer, respectively, and $y$ is the corresponding binary classification prediction result, which is mapped to the range of $[0,1]$ by the Sigmoid function.

D. Fusion of Model

Assume the outputs of the model are $o_1$ and $o_2$ and the hyperparameter $k$, then the final probability distribution is computed as follows:

$$\text{output} = k \otimes o_1 + (1 - k) \otimes o_2 \quad (9)$$

where $k$ is normally equal to 0.5. For both models, we all use binary cross-entropy loss to optimize.

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

Fig. 2: Overview of our DSIR method.
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. EXPERIMENTS

#### A. Dataset Preparation

First, We utilize the same dataset of 500 high-quality Java projects from GitHub as used in [9], covering a variety of functions and diverse application areas. Then we construct a dataset of UML class diagrams for the packages in each project by UMLGraph\(^2\). Designite [19] is used to detect code smells in our dataset and generate corresponding smell reports. Finally, we select three types of design smells based on their frequency of occurrence in the 500 high-quality Java projects and label the corresponding class-level code fragments accordingly. We divide projects into three parts, 70% as the training set, 10% as the validation set, and 20% as the test set. Moreover, we carefully balance the dataset to ensure that samples from the same package are not split into different sets and reduce the number of negative samples to the balanced distribution of samples. Table I presents the number of samples used in our DSIR method, as well as the baselines.

<table>
<thead>
<tr>
<th>Code smells</th>
<th>Training set</th>
<th>Validating set</th>
<th>Testing set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broken Hierarchy</td>
<td>5000</td>
<td>369</td>
<td>1628</td>
</tr>
<tr>
<td>Insufficient Modularization</td>
<td>1500</td>
<td>137</td>
<td>420</td>
</tr>
<tr>
<td>Deficient Encapsulation</td>
<td>5000</td>
<td>881</td>
<td>1899</td>
</tr>
</tbody>
</table>

#### TABLE I: Sample distribution of our DSIR dataset

#### D. Configurations

Fig 3 presents the performance of the DSIR method under different configurations, including learning rates, batch sizes, and model settings. Based on different performance, we seek the best configuration. Our DSIR model consists of two components: R-GCN and LSTM. For R-GCN, we set the embedding dimensions to be 512 and the number of hidden units to be 300 for Insufficient Modularization and Deficient Encapsulation. For LSTM, we set the embedding dimensions to be 200, the hidden dimensions to be 200 for Broken Hierarchy and 150 for Insufficient Modularization and Deficient Encapsulation. For the first sub-model, we apply dropout with a rate of 0.4 to prevent overfitting. For ASTNN, we set two layers and 128 dimensions in the hidden dimension layer and 256

---

\(^2\)https://github.com/dspinellis/UMLGraph
TABLE II: Performance of DSIR and other baselines.

<table>
<thead>
<tr>
<th></th>
<th>Broken Hierarchy</th>
<th>Insufficient Modularization</th>
<th>Deficient Encapsulation</th>
<th>Avg-F1</th>
<th>Δ(↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
<td>AUC</td>
<td>P</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.36</td>
<td>0.63</td>
<td>0.46</td>
<td>0.59</td>
<td>0.22</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.17</td>
<td>0.74</td>
<td>0.27</td>
<td>0.66</td>
<td>0.18</td>
</tr>
<tr>
<td>ASTNN</td>
<td>0.52</td>
<td>0.48</td>
<td>0.50</td>
<td>0.69</td>
<td>0.64</td>
</tr>
<tr>
<td>DISR -w/oR-GCN</td>
<td>0.59</td>
<td>0.58</td>
<td>0.58</td>
<td>0.73</td>
<td>0.86</td>
</tr>
<tr>
<td>DISR -w/oLSTM</td>
<td>0.78</td>
<td>0.60</td>
<td>0.68</td>
<td>0.75</td>
<td>0.47</td>
</tr>
<tr>
<td>R-GCN -w/oClass-relation</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.65</td>
<td>0.40</td>
</tr>
<tr>
<td>R-GCN -w/oClass-info</td>
<td>0.42</td>
<td>0.62</td>
<td>0.50</td>
<td>0.65</td>
<td>0.17</td>
</tr>
<tr>
<td>DISR</td>
<td>0.76</td>
<td>0.63</td>
<td>0.69</td>
<td>0.76</td>
<td>0.87</td>
</tr>
</tbody>
</table>

encode dimensions. Then we choose 80 features and 50 trees in the random forest. Additionally, we use the Adam optimizer algorithm with a 0.002 initial learning rate and the batch size is set to be 16.

V. PERFORMANCE OVERVIEW

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

We conduct experiments on our datasets and other baselines. As shown in Table II, DSIR outperforms other baselines, including the state-of-the-art ASTNN, and achieves a 31% improvement in F1-measure, which confirms our initial intuition and demonstrates the effectiveness of our method. Lack of inter-class relation and AST node tokens, other baselines fail to give full play to the same excellent performance as implementation smells in the face of design smells. Specifically, design smells with more complex and longer code snippets will lead to the loss of features for the entire sub-tree if we truncate long traversal sequences of AST. Furthermore, we apply 80 metrics [10] to Random Forest and low AUC values can be observed for all three design smells. We believe that the reason for this is that the complex information of design smells makes it difficult for algorithms to construct rules for detection.

TABLE III: Mappings between Cliff’s Delta values and their effective levels.

<table>
<thead>
<tr>
<th>Cliff’s delta</th>
<th>Effective levels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>δ</td>
</tr>
<tr>
<td>0.147 ≤</td>
<td>δ</td>
</tr>
<tr>
<td>0.33 ≤</td>
<td>δ</td>
</tr>
<tr>
<td>0.474 &lt;</td>
<td>δ</td>
</tr>
</tbody>
</table>

TABLE IV: Win/Tie/Loss indicators on F1 values of Random Forest, ASTNN, LSTM and DSIR.

<table>
<thead>
<tr>
<th>Code smell</th>
<th>DSIR vs Random Forest</th>
<th>DSIR vs ASTNN</th>
<th>DSIR vs LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broken Hierarchy</td>
<td>&lt;0.05(+Large)</td>
<td>&lt;0.05(+Large)</td>
<td>&lt;0.05(+Large)</td>
</tr>
<tr>
<td>Insufficient Modularization</td>
<td>&lt;0.05(+Large)</td>
<td>&lt;0.05(+Large)</td>
<td>&lt;0.05(+Large)</td>
</tr>
<tr>
<td>Deficient Encapsulation</td>
<td>&lt;0.05(+Large)</td>
<td>0.05(+Large)</td>
<td>&lt;0.05(+Large)</td>
</tr>
<tr>
<td>Win/Tie/Loss</td>
<td>3/0/0</td>
<td>3/0/0</td>
<td>3/0/0</td>
</tr>
</tbody>
</table>

Finally we apply the Win/Tie/Loss indicator as a common approach in prior works for performance comparison [11], [12]. We also conduct Wilcoxon signed-rank test and Cliff’s delta test to further analyze the performance of our model and baselines. Table III shows Cliff’s delta values(|δ|) and the corresponding effective levels. We use a comparison method to determine the Win/Tie/Loss indicator as follows: First, we select a baseline method M. If our model outperforms M with a p-value < 0.05 and a Cliff’s delta ≥ 0.147, we mark our model as a “Win” indicating a statistically significant difference. Conversely, if the baseline method M outperforms our model with a p-value < 0.05 and a Cliff’s delta ≥ 0.147, our model is marked as a “Loss”. Otherwise, we mark it as a “Tie”.

According to the results shown in Table IV, our model performs significantly better than all the compared models in detecting design smells, demonstrating excellent effectiveness.

B. RQ2: What impact does each of our main components have in our model?

To evaluate the impact of each main component in our model, we conduct an ablation study by comparing the performance of two individual models and their final fusion model. The results of the two models in Table II show that the R-GCN model performs better on code smells like Broken Hierarchy and Insufficient Modularization, while the LSTM model performs better on code smells like Deficient Encapsulation and Insufficient Modularization. However, each model has its own limitations and neither of them can effectively capture all the features of comprehensive features of code smells. Meanwhile, R-GCN does not work well for Deficient Encapsulation because Deficient Encapsulation focuses more on the encapsulated information inside class members, so the LSTM using AST tokens can better extract the details of the class members, but this still requires information from the class relation to determine if these class members should be encapsulated through inter-class relation.

C. RQ3: How does the class-diagram contribute to our proposed model?

In Section III, we introduce the class diagram consisting of two parts: extracting complex relation between classes and extracting internal information of each class. To investigate their impact on our DSIR method, we conduct the following experiments. First, we evaluate the impact of different types...
of relation between classes by comparing the performance of a Graph Convolutional Network model trained on the same dataset but not distinguishing between edge types. Then, we evaluate the impact of class information in the class diagram by removing method and attribute information from the previously mentioned tuples and only using the class name as a feature.

As shown in Table II, different types of relation between classes can effectively help the model learn the external hierarchy of classes and the interactions between modules. The experiment without relation types results in a 28% decrease in F1 score for Broken Hierarchy and a 47% decrease in F1 score for Insufficient Modularization. However, for Deficient Encapsulation, the relation types do not have a significant effect, as this smell focuses more on the encapsulation information of class members. In this case, class methods and attributes are more useful for understanding the internal features of classes in complex relations, providing encapsulation information and improving efficiency. Moreover, the information of methods and attributes as class internal features can also help the model better understand issues such as function confusion or complex responsibilities within classes, which are associated with Broken Hierarchy and Insufficient Modularization.

In summary, the inclusion of different types of class relation and internal information of each class in class diagrams enhances the model’s discriminative and expressive capabilities, thus improving the performance of code smell detection.

VI. THREATS TO VALIDITY

A. Internal validity

We utilize the UMLGraph tool to construct a class diagram dataset and remove projects where class diagrams could not be automatically generated due to tooling issues. This may introduce bias if these projects have different characteristics than those in our dataset. Meanwhile, we use the Designite tool to label our training data as ground truth. Although the tools are widely used, their reliability still needs to be verified.

B. External validity

We only use Java projects from GitHub, which may limit our ability to extend our findings to other programming languages or domains. Additionally, the limited number of positive and negative samples and the random reduction of negative samples to match the positive samples may introduce sampling bias if the sampled projects have different characteristics than the population of Java projects.

VII. CONCLUSION

In this paper, we propose a new approach for detecting design smells, namely Design Smell Detection through Inter-class Relation. We apply the AST node tokens instead of code tokens or AST, and inter-class relation to design smell detection, which are separately fed to BiLSTM and R-GCN networks to extract the comprehensive features. Extensive experiments are conducted on a new dataset, and the results demonstrate that the proposed method outperforms state-of-the-art methods in terms of F1-measure of all code smells.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (62172202), Collaborative Innovation Center of Novel Software Technology and Industrialization, the Major Program of the Natural Science Foundation of Jiangsu Higher Education Institutions of China under Grant Nos. 22KJA520008, the Priority Academic Program Development of Jiangsu Higher Education Institutions, and the Undergraduate Training Program for Innovation and Entrepreneurship, Soochow University (202210285193H).

REFERENCES

Dissecting Scale-Out Applications Performance on Diverse TLB Designs

Tianmai Deng, Yixiao Xu, Chi Zhang, Zhengwei Qi*
Shanghai Jiao Tong University
Shanghai, China
{tianmaideng, xuyixiao.2019, sjtu-zhangchi, qizhwei}@sjtu.edu.cn

Abstract—Scale-out applications, such as various big data systems and memory computing programs comprise an important software stack in clouds. Such applications usually have large memory data footprint as well as code sizes, thus stressing the CPU’s TLB efficiency. In this paper, we experimentally evaluate how various TLB design choices in modern off-the-shelf x86 CPUs impact the performance of scale-out applications. The findings aim to guide the partitioning schemes and capacity planning of TLBs, and software-hardware co-design for emerging applications.

Keywords—performance evaluation, scale-out applications, TLB structures, MMU overhead, TLB miss rates

I. INTRODUCTION

In cloud service environments, dominant scale-out applications [1], such as Hadoop and Spark for distributed data analysis, and back-end applications like Nginx and MySQL for web services, are widely deployed. These applications exhibit markedly different memory characteristics at runtime compared to traditional Linux applications. In particular, the executable binary file size and the data working set size [2] of scale-out applications have significantly increased. This poses unprecedented memory access pressure [3] on non-supercomputing data center machines.

At the same time, as the main frequency increases and pipeline efficiency improves [4], CPUs favor continuous feeding of instructions to maximize their processing power. However, modern scale-out applications tend to issue a large body of load/store operations that stress the MMU for address translation. Address translation in modern 64-bit CPUs is non-trivial, which contains 4-layer page walks along with cache misses for PTE accesses. Thus, TLBs are indispensable components to mitigate such expensive memory operations.

For high-performance scenarios on modern CPUs, TLB entries should be filled only once (initial translation) and held for the entire application lifecycle. In such cases, all memory accesses within the application avoid address translations and exhibit the same TLB access latency, which is fast due to the hardware nature of TLB. However, in real-world scenarios, TLB misses still happen [5] and damage application performance due to bad data locality, code and data contention of TLB entries, or fragmented address accesses.

Previous works conducted performance analysis on data center applications, such as middleware [6] and serverless frameworks [7]. Especially, Michael et al. [5] analyzed the MMU overhead for scale-out applications, which can account for 1.8% to 16% of its runtime. However, such overhead is by far not well understood with regard to the specific TLB designs in modern CPUs. This paper aims to fill this gap and advance the understanding.

To investigate the impact of TLB design on scale-out applications performance, this paper conducts experiments on different processors, including Intel’s 9th and 12th generation processors and the AMD Zen3 processor, which use the same x86 ISA but embody very different TLB designs. We analyze the performance of applications and measure the miss rate of TLB, under various workloads and system huge page configurations. Specifically, we explore various TLB design details and clarify how different TLB capacities and structures affect the scale-out application performance.

The main contributions of this paper include:

- We find that some scale-out applications still suffer from a non-negligible data memory performance overhead of up to 10.9% due to TLB misses.
- Applications with small hot code areas (such as data-caching, in-memory-analytics, and graph-analytics) have reduced code performance loss to less than 0.1% on the latest CPUs.
- Applications with store-major memory operations (such as memcached and redis) tend to have better performance when the underlying CPU has a unified dTLB for loads and stores.
- Applications with frequent accesses to a small part of hot data will be greatly impacted by the L1 dTLB latency. Thus, an L1 dTLB with higher associativity (thus lower latency) combined with extensive uses of huge pages (manual application code tuning or OS-level allocator policy) can effectively improve their performance.

The rest of the paper is organized as follows. Section 2 introduces the potential performance impact of paging and TLB design on scale-out applications. Section 3 presents our experimental setup and evaluation methodology. Section 4 discusses the results of our experiments, and presents the performance analysis and discussion. Finally, Section 5 concludes the paper.

DOI reference number: 10.18293/SEKE23-139
II. PERFORMANCE IMPACT OF TLB

This section illustrates that the paging memory system is not suitable for scale-out applications, and describes several TLB designs and their impact on the performance of different applications.

A. TLB, the legacy of paging memory

The Memory Management Unit enables efficient memory allocation and management through paging. In the early days of the computer, the need for memory performance in applications was not yet as great as the demand for memory capacity [9], so the more flexible and efficient memory utilization that paging brings outweighs the additional access latency when an address translation occurs. Nowadays, 64-bit CPUs are widely used where 4-layer page tables replace the 2-layer tables on 32-bit CPUs and incur a 2x latency penalty. Also, the main frequency of CPUs has greatly increased over time. The CPU has to wait for more cycles when memory access stalls due to address translation.

However, scale-out applications are placing higher demands on memory capacity [3], and the additional logic introduced is causing the binary file size to balloon [2]. In addition, the working set size has grown geometrically to handle the ever-increasing amount of data. The scale-out applications demand far more performance and capacity from both code and data memory than they did when paging was proposed, and the paging design with flexibility is no longer performant for scale-out applications.

TLB [9] and huge page [10], as mitigants of the paging memory system, are unable to fully resolve the divergence between the scale-out applications and the paging memory system. Every TLB miss can result in applications being exposed to the paging memory system and having to endure meaningless page walk latency [11]. Therefore, how to utilize TLB resources as much as possible on top of the decaying paging memory system to avoid negative performance impact on scale-out applications is a matter of concern.

B. Impact of TLB Structures

Different TLB designs are not oriented to the same memory access pattern and may exhibit different performance and utilization for different applications and huge page setups.

In the L1 TLB, AMD and Intel adopt two different implementations of TLB entry for various page sizes, which we call generic entry and specific entry, as illustrated in Figure 1. Generic entry design [8] is that each L1 TLB entry can store address translation results for all page sizes. However, this design limits the number of TLB entries due to the need for recording more bits per entry. In contrast, the specific entry design [12] stores 4K pages and 2M huge pages in separate entries in the L1 TLB. However, the number of entries reserved for huge pages is limited, which reduces the effective range of address mapping when using huge pages (Table 1).

There are also two different approaches for L2 TLB. Intel’s approach [8] is combining iTLB and dTLB in the merged L2 TLB, also known as the sTLB, to record address translation results for both code and data memory. Although TLB entry updates may face contention, this design allows for more flexible capacity sharing. AMD chooses to separate the sTLB into distinct L2 iTLB and L2 dTLB [12] to store the address translation results for code and data memory, respectively, without interference. However, this approach may not fully utilize all hardware resources when the pressure of the two types of memory is unbalanced.

III. METHODOLOGY

This section describes the test load selection, test environment setup, and test methods used in our experiments.

A. Simulated Load

CloudSuite benchmark [13] is a set of test applications or collections for different scenarios that cover various situations that may occur in data centers. To measure the performance of scale-out applications, we used CloudSuite benchmark 4.0 as a simulated data center workload in our study.
B. Test Environment Setup

To compare the performance impact of different TLB capacities and structures, we set up the experimental environment on three hosts with different processors:

- Host A: 6-core Intel Core i5-9600K (Coffee Lake) CPU running at 3.70GHz with DDR4 32G memory.
- Host B: 8-core Intel Core i7-12700 (Golden Cove) CPU running at 4.50GHz with DDR4 16G memory.
- Host C: 8-core AMD Ryzen 5700G (Zen3) CPU running at 4.60GHz with DDR4 32G memory.

The detailed TLB specifications are recorded in Table 1. All hosts run the Debian 11.6 system with Linux kernel 6.0.12-1.

C. Performance Event

To analyze the running characteristics of the same workloads on processors with different TLB capacities and structures, we bind client and server applications to different cores, while only measuring the cores running the server application. To collect processor performance events during testing, we employed the Linux perf tool, and the average of three test results was computed.

D. Huge Page

If only 4K pages are used, the TLB can provide an extremely limited range [14] of address mapping (Table 1), which is much smaller than the size of the scale-out application’s working set, or even the size of its code memory, but the use of huge pages can effectively alleviate this limitation.

In Linux, there are two ways to use huge pages: Transparent Huge Pages (THP) and hugetlbfs. THP is transparent to the application and does not require modification of the application, while hugetlbfs is not. And starting from kernel 5.14, THP can not only be used for data but also executable memory, mapping application code areas with huge pages. However, if the memory pages are not aligned to 2MB or not contiguous, the promotion rate of THP will not only decrease further but also introduce noticeable memory bloat [15].

IV. EXPERIMENT RESULT AND ANALYSIS

In this section, we analyze application performance overhead on processors with various TLB designs. Then, we investigate the effects of using huge pages. Additionally, we analyze the importance of L1 iTLB and L1 dTLB for applications with limited hot code areas and frequent hot data accesses. Furthermore, we explored the impact of a separate store dTLB on data memory performance. Finally, we measure the effective hit rate of L2 TLB on different structures and whether it matters to applications.

A. Is MMU overhead still significant for scale-out applications on the latest x86 processors?

In this section, we measure and calculate the proportion of cycles consumed by page walks (Host A and Host B) or pipeline stalled cycles (Host C) during application runtime to evaluate the MMU overhead caused by TLB misses for different scale-out applications. Figure 2 shows the results.

For most scale-out applications, the performance overhead due to iTLB misses is not significant. The average iTLB misses overhead is only 0.98%, 0.56%, and 0.35% on Host A, Host B, and Host C. However, for applications with large hot code areas such as data-serving and web-serving, the iTLB miss overhead is still non-negligible, especially on processors with small TLB capacities such as Host A, where the overheads are 3.89% and 0.86%, respectively.

Since the MMU overheads are not comparable due to different performance event definitions between Host C and the others, we analyze the TLB miss rates (Figure 3 and Figure 4). We find that the L2 iTLB miss for most applications is less than 0.25 MPKI, for applications with large hot code areas it still does not exceed 1 MPKI. However, the L2 iTLB miss rate of these scale-out applications on Host C is on average 63.26% higher than that on Host B, which suggests that applications running on Host C may get worse code performance.

Moreover, by comparing the code performance overhead on Host A and Host B, we find that Host B has indeed alleviated the code memory address translation overhead by increasing the iTLB and sTLB capacity. The code performance overhead on average decreases by 39.61%, and in data-caching the overhead remarkably decreases by 98.2%.

In contrast to the low code performance overhead, our results show that most scale-out applications still suffer from significant data memory performance overhead due to dTLB misses. Specifically, these applications have an average overhead of more than 3.68% on Host A and Host B, and applications with higher data memory pressure, such as data-caching and data-serving, introduce up to 11.71% and 7.33% overheads.

Findings:

- In scale-out applications, the overhead caused by iTLB misses is not significant, with most applications showing a code performance loss of less than 1%.
Most scale-out applications still suffer from a non-negligible data memory performance overhead. The dTLB miss overhead on the data-serving application exceeds 10% on older processors, which can be reduced by over 25% by enlarging dTLB capacity.

In database applications and others with large hot code areas, the iTLB miss overhead is still noticeable and should be given due attention.

Separating the L2 iTLB from the L2 TLB has a non-efficient improvement on the overall application code performance.

B. Can huge pages improve the code and data memory performance of applications?

The comparison of MMU overhead before and after enabling THP reveals that using huge pages for code and data memory leads to better overall performance, resulting in an average of 12.16% reduction in iTLB miss overhead and 27.43% in dTLB miss overhead. For data-serving applications, enabling THP leads to a significant reduction in iTLB miss overhead by an average of 23.30%, resulting in better code performance. Additionally, for data-analytics, in-memory-analytics, and graph-analytics applications, using huge pages leads to a reduction in data memory performance overhead by 48.49%, 58.73%, and 56.45%, respectively. Even for web-serving databases that are not suitable for huge pages [16], enabling THP still reduces iTLB and dTLB miss overhead by 12.58% and 18.41%. However, it is important to consider the potential long-tail in memory access latency due to THP memory page locking [15], and explicit use of huge pages is recommended instead.

Findings:

- In most scale-out applications, even those with sparse data memory access patterns like database applications, enabling huge pages has a significant improvement in both code and data memory performance.

C. What can be observed when running on processors with different TLB designs?

We analyze how different TLB designs affect application performance. Figure 4 illustrates the L1 iTLB and L2 dTLB miss rate on different processors. For every application, Host B outperforms the others, with an average 65.28% and 59.47% reduction in iTLB miss rate to Host C and Host A.

In data-caching, in-memory-analytics, graph-analytics, and web-search applications, Host B has a lower L2 dTLB miss rate than Host A, but with a higher data memory performance overhead (Figure 2). One possible cause of this phenomenon is that the Linux kernel swaps out [17] some pages to disks due to limited system memory capacity, which causes orders of magnitude data memory performance degradation of the program when accessing these pages.

For code performance, except for applications with limited hot code areas, such as data-caching and media-streaming, Host C’s fully associative design of L1 iTLB is not as effective as larger but less associative L1 iTLB on other applications. The L1 iTLB miss rate of Host C is 14.33% higher compared to Host A. Host B continues to increase its capacity while keeping its associativity unchanged, further reducing the average L1 iTLB miss rate by 59.47%. This is particularly evident in data-caching, an application with limited hot code areas, where the L1 iTLB miss rate of Host B is reduced by 63.67%.

Regarding the huge pages, we found that enabling THP resulted in reduced iTLB and dTLB miss rates for all three processors in most applications, with an average reduction of 14.21% and 46.72%, respectively. And the data in Figure 3 shows that enabling THP also led to a reduced average L2 iTLB miss rate of 22.52%.

Lastly, we observed an abnormal increase in the L1 iTLB miss rate of the data-serving application on Host B when enabling THP, with an increase of 20.03%. However, a similar phenomenon does not occur on Host A, which has fewer iTLB
entries for huge pages. This may be attributed to the iTLB of Host B has a huge capacity for 4K pages and can perfectly fit the hot code areas of the application, while not limited by the limited number of huge page entries.

Findings:
- No matter which processor and what TLB design is considered, enabling huge pages can lead to an overall improvement in application performance.
- For scale-out applications with limited hot code areas, the impact of L1 iTLB capacity has reached a critical point. Running on processors with larger L1 iTLB can almost eliminate the address translation overhead on code performance.

D. How does the design of L1 dTLB affect centralized hot data memory accesses?

Centralized memory accesses to a small part of hot data will be greatly impacted by the L1 dTLB design. In VM-based cloud environments, nested page tables make the address translation overhead even higher [18]. For applications with frequent accesses to hot data that may be sensitive to latency, it is necessary to ensure that all memory access hit the TLB to avoid data memory address translation.

Figure 5 shows the miss rates of L1 dTLB. For all scale-out applications, it can be observed that Host C’s L1 dTLB performs significantly better than Host A and Host B. The average L1 dTLB miss rate for Host C is only 3.2487 MPKI, which is 47.39% and 36.73% lower than Host A and Host B, respectively. Given the similarity of the L1 dTLB capacity among the three hosts, Host C’s fully associative design is likely the reason for its higher hit rate.

After enabling THP huge pages, the L1 dTLB miss rates on Host C, Host A, and Host B are reduced for any applications. The average reduction is up to 58.52%, indicating that using huge pages is also extremely useful for reducing the hot data memory access latency.

Findings:
- For applications requiring low latency access to hot data, a processor with higher associativity on L1 dTLB should be considered.
- The use of huge pages effectively reduces the L1 dTLB miss on different processors, which further reduces access latency on hot data access.

E. Do hybrid load/store applications suffer a performance hit due to store dTLB?

We noticed that in Intel’s latest dTLB design, the store operations have been allocated a small separate area. Therefore, we separately measured the hit rates of load and store accesses in the L1 dTLB, as shown in Figure 6.

Firstly, for all applications running on Host B, the miss rate of the store operation in L1 dTLB is significantly higher than that on Host A, with an average increase of 129.73%. However, by avoiding store operations from competing with load operations for limited TLB resources and mitigating the interference with the address translation of load operations, the TLB miss rate of load operations is dramatically reduced by 44.57%. This further contributes to a 22.22% decrease in the overall miss rate of the L1 dTLB, thus providing better data memory performance for applications.

However, for applications with store-major memory operations, such as data-caching, the store dTLB miss rate increased by 187.55%, despite a decrease of 29.27% in the load L1 dTLB miss rate. The high store miss rate leads to overall performance degradation of the L1 dTLB. Nonetheless, since most applications mainly issue load operations, the separate store dTLB does help reduce the dTLB miss rate without noticeably increasing the dTLB capacity.

Findings:
- For applications that have a high proportion of store operations, running on processors without separated dTLBs can probably provide better overall performance.
has reached a critical point where almost all code address with limited hot code areas, the current L1 iTLB capacity loads and stores. Noteworthy, we found that for applications requirements, such as frequent hot data accesses or mixed a processor with the appropriate TLB design can maximize TLB misses. When running scale-out applications, selecting degradation of code and data memory performance due to capacity and associativity, they still cannot fully mitigate the to the paging memory system and address translation. We overhead suffered by a variety of scale-out applications due to paging by incorporating in-depth analysis on TLB design, and we hope the findings can guide the planning of TLBs and CPUs, and software-hardware co-design for the next generation of applications.

VI. ACKNOWLEDGEMENT

This work was supported in part by National NSF of China (No. 62141218), and Shanghai Key Laboratory of Scalable Computing and Systems.

REFERENCES


Real-time Estimation Approach to Scrum Projects Effort: Task Point and Individual Reliability

Yu Guo, Jun Guo*, Mohan Shi, Aixuan Dong
School of Software College
Northeastern University
Shenyang, China

Abstract—In software project management, it is a useful method to control the progress of the project through effort estimation. How to estimate the effort in real time in the Agile software development has always been a difficulty faced by Agile teams. Aiming at the current mainstream Agile Software Development techniques, Scrum framework, this paper puts forward the calculation approach for reliability of Scrum team members and the Sprint development time estimation approach based on individual reliability, i.e., the total working time estimation of the next Sprint is realized by estimating and adjusting the working time of the team individuals. 51 development teams with 5-9 people have put this method into practice for experimental verification, and the results show that the highest estimation accuracy of this approach can reach 84.1%.

Keywords- Scrum; individual reliability; task point; effort estimation

I. INTRODUCTION

Agile software development is a software development method with strong adaptability and flexibility. There are many Agile development methods (e.g., Extreme Programming (XP), Agile Unified Process (AUP), Scrum), favored by enterprises. Scrum is one of the representative approaches [1]. In software project development, making a comprehensive development plan is the foundation of project success [2]. In order to achieve this goal, the team needs to estimate the effort of the project. Software project effort estimation refers to with the analysis of business objectives, considering current environmental conditions, human resources and other factors, the team estimates the time required for the completion of different stages of the project or the time required for the completion of the whole project [3]. Basing the estimation result, the team members can make a schedule for the completion time of their tasks, which is conducive to supervise the team members’ work and effectively manage the software project.

It is one of the representative approaches of traditional software project effort estimation to estimate through function points [4]. This approach was first proposed by Albrecht, and IFPUG appeared later, developing Albrecht's idea. Function point is a function divided according to user requirements in software project. IFPUG realizes the effort estimation with the support of the quantified result of the input, output and file type of all function points and the adjustments of the estimation adjustment factor [4] [5]. With the further studies of researchers, approaches such as COSMIC were proposed to modify the deficiencies of IFPUG [6] [7] [8]. Although the traditional effort estimation approaches have gradually been more and more mature, it is hard for them to adopt the unpredictability and huge changeability of requirements nowadays [9].

As time goes by, software development framework has gradually improved to deal with these problems. Agile software development framework represented by Scrum is the mainstream currently. Native Agile effort estimation methods focus on the thoughts of each team member, which is highlighted in the estimation methods such as Story Points, Poker Planning, Fibonacci Sequence and T-shirt Size. Many studies on Agile estimation researched the characteristics and shortcomings of these methods. In addition, some researches put forward new methods to solve these problems. In [10][11], machine learning was used for estimation. What’s more, the authors in [10] compared machine learning, expert opinions and mathematical algorithm models, finally concluding that machine learning is more accurate in Agile estimation; By analyzing the text of user story description, Sudarmaningtyas and Mohamed [10] got 83% accuracy of story point estimation. In [12], the effort of Scrum-based IT projects was estimated with the usage of fuzzy logic model. Arora et al. [13] introduced the adaptive neuro-fuzzy inference system (ANFIS) and the novel Energy-Efficient BAT (EEBAT) technique to effort estimation in Scrum, by comparing state-of-the-art meta-heuristic and machine learning with ANFIS-EEBAT approach, concluding that ANFIS-EEBAT performs best. The one by Ramesur and Nagowah [14] used machine learning technique, considering 12 factors that have an influence on effort estimation, to predict a Sprint work time. Butt et al. [15] provided the knowledge which factors may lead to low accuracy of effort estimation in Scrum environment and the framework to avoid the bad impact. Authors in [16] mainly studied good and bad practices in the Agile software development projects, contributing to finding the key to correct estimation. And Almeida et al. [17] holds the opinion that the metrics related to the business value delivered and the success of sprint goals are the most relevant. They also investigate the correlation of other metrics to Scrum process monitoring.

Traditional methods estimate the effort of software development in the stage of requirements analysis, which has great limitations in the changeable and volatile software development environment. Thus, the accuracy of estimation is very poor [18]. Furthermore, in the existing researches on Agile estimation methods, some put forward metrics that have an impact on the accuracy of estimation, while do not give a specific improvement scheme [16][17]; and some use theories

DOI reference number: 10.18293/SEKE23-001
or models such as machine learning, natural language processing, etc. [10][11][12][13][14] for effort estimation, which do not embody the characteristics of Agile in the models to a certain extent.

Therefore, we propose a novel concept of task point for Scrum-based IT projects, based on the concept of the story points estimation method. Task point measures the effort of individual in the team, and then the effort estimation of the whole team can be obtained according to individual task points. What's more, for the sake of improving the accuracy of estimation, the reliability of individual is considered to adjust the estimation result. After each Sprint, the reliability of individual will be adjusted and applied to the next effort estimate. The advantage of this approach is that it not only refines the estimation granularity of Scrum, but also takes full account of the reliability of individual estimation, which can help gain the effort estimation more accurately in real time in every Sprint, thus making the approach high availability.

II. IDEAL TIME AND TASK POINT

In Scrum, each user story is closely related to the requirements in the project backlog. Project development serves user stories, so estimating the ease of development in Scrum usually requires estimating the ease of completing each user story. In that case, Story Point [19] is put forward for the purpose. We don't use specific units, such as man-months, to express the effort, but use the relative concept of story point to estimate the difficulty of user stories. It is commonly believed that team decisions are more significant in Scrum, so the story point of each user story is obtained by integrating the opinions of team members. That's the reason why ideal time and task point are proposed. Task point can be seen as the evolution of story point. In practice the effort measured by time is easier to understand. Therefore, the ideal time is proposed as the basic unit of task point size, and the task point is the individual's estimation of completing a task.

A. Ideal Time

Ideal time is the time to complete the work excluding other activities unrelated to the work. Its attributes are as follows:

1) Uniqueness: The user story task being estimated is the only work that the people needs to do.
2) Self-containment: When the task starts executing, everything needed is ready, independent of other tasks.
3) Independence: There is no interference in the process of doing it.
4) Generality: The efficiency of the person performing the task is normal.

B. Scrum Task Point

Task point is the amount of time a task takes, corresponding to the effort of the task in Scrum. During development, the basic unit of task point is ideal time.

The team's development consists of multiple Sprint processes in Scrum. In a Sprint cycle, the team will solve 1 ~ n user stories, and each story will be broken down into 1 ~ n tasks to be completed by different developers. After the Product Owner makes a backlog and each team member chooses tasks, the development work begins. Before a Sprint starts, team members need to estimate the ideal time required to complete every task, then the time recorded as task point.

By calculating the task points of all tasks contained in a user story, it is easy to determine the time required for the development of the user story. And the time of all user stories in this Sprint is the development time of this sprint.

III. REAL-TIME EFFORT ESTIMATION APPROACH BASED ON SCRUM TEAM MEMBER RELIABILITY

In most of the methods based on experts to estimate effort, the experts’ knowledge and development experience play an important role in estimating and controlling the progress of different periods of the project. However, the reality is that experts may ignore the overall level of the team or other factors when estimating, which makes the estimation result meaningless [20]. In the method described in this paper, we pay more attention to the estimation data of team members to complete tasks and their estimation accuracy. It is obvious that the estimation results based on the real level of team will be more reasonable.

A. Calculation Method for Scrum Team Members’ Reliability

Before Scrum-Based IT project development, the team will estimate the story points of user stories. The common practice is to take the development difficulty of a certain user story as the baseline. Then the story points of other user stories will be determined relative to the baseline. Inspired by it, we propose a brand-new calculation method to estimate the time when the user story will be completed. In this method, the ideal time for members to complete tasks and the accuracy of individual estimation are the basis of calculation. After members pick up tasks, they estimate the completion time of each task. Assuming that the ideal time for a task to be completed is \( x \). After the member completes the task, the actual time taken is \( y \). Afterwards, the reliability of this task \( R \) is

\[
R = \frac{y}{x}
\]

When the \( n \)th task is finished in this sprint, the reliability of this task will be compared with the maximum and minimum values of the 1st ~ \( n-1 \) task reliability of this member. If it is greater than the maximum value, the maximum value will be changed to this value. While if it is smaller than the minimum value, the minimum value will be updated.

B. Development Duration Estimation for Sprints Based on Individual Reliability

Having the definition of individual reliability and its calculation method, we can make effort estimation of a Sprint. During the \( \text{th} \) round of Sprint, each member is assigned to \( n \) tasks. Before the software development begins, a task completion recording table for each one will be established. Table I shows how to record members’ task completion.
TABLE I. MEMBER TASK COMPLETION RECORDING TABLE

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Expected Working Time</th>
<th>Actual Working Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>$a_1$</td>
<td>$a_1'$</td>
</tr>
<tr>
<td>Task 2</td>
<td>$a_2$</td>
<td>$a_2'$</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Task n</td>
<td>$a_n$</td>
<td>$a_n'$</td>
</tr>
</tbody>
</table>

The member primarily estimates the expected completion time of each task. For example, for task $i$ the expected working time is $a_i$. After it is completed, the actual working time would be recorded as $a_i'$. With $a_i$ and $a_i'$, we can get this reliability of task $i$ is $R_i$. After that we compare the reliability maximum $R_{\text{max}}$ and minimum $R_{\text{min}}$ of this member with $R_i$ respectively to update the reliability range.

After the team has completed all the development of the $i$th Sprint, we can determine the reliability range of each member in this Sprint. Then we compare the individual reliability range of $i$th Sprint $r_i$ with the reliability of $i$-1th Sprint $r_{i-1}$, if $r_i \leq r_{i-1}$, the reliability of this person will be updated to $r_i$. Otherwise, we should compare the reliability boundary values of the two Sprints respectively, i.e. the minimum boundary value of the latest reliability is the smaller value of the minimum boundary values between $r_{i-1}$ and $r_i$ and the maximum boundary value of the latest reliability is the larger value of the maximum boundary values between $r_{i-1}$ and $r_i$. After updating the individual reliability, it will be used to estimate the time of the next Sprint. At the beginning of the $i+1$th Sprint, members still give their own estimation time for each task, and then we can use their latest individual reliability to adjust and get their own range of task completion time. If a member's reliability range is $[R_{\text{min}}, R_{\text{max}}]$, and his estimation time for task $m$ is $a_m$. Then the completion time of task $m$ will be estimated as (2):

$$T_m = \begin{cases} t_{\text{max}} = R_{\text{max}} \times a_m \\ t_{\text{min}} = R_{\text{min}} \times a_m \end{cases}$$ (2)

With the continuous adjustment of reliability, the estimated completion time of each task to be completed is constantly changed, and the predicted completion time of the $i+1$ Sprint also gets persistently adjusted. Furthermore, the effort estimation can be visually displayed in the Scrum Kanban, thus making the team development process more transparent. Considering the result of this approach is a time range, it also gives the team a buffer in the development, which is beneficial for the team to allocate the work more reasonably in real time and improve the development efficiency.

IV. EXPERIMENTAL VERIFICATION

A. Experimental Process

In order to validate the methodology presented, we assigned 51 small student teams in Northeastern University to develop the same Scrum-Based IT project. Before the project development, the product backlog and the story point of each user story has been determined. This project is divided into three Sprints. In each Sprint, All the teams record the estimated work time and the actual work time of each task, calculating the reliability of each person in the development project. We continue to track the development process of each team and investigated the changes of the accuracy of team effort estimation in each Sprint.

The change of everyone’s reliability in a 6-people team in three rounds of Sprint is shown in Table II:

TABLE II. MEMBER RELIABILITY CHANGE

<table>
<thead>
<tr>
<th>Members</th>
<th>Reliability Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sprint1</td>
</tr>
<tr>
<td>Member1</td>
<td>1.0669</td>
</tr>
<tr>
<td>Member2</td>
<td>0.98-1.0412</td>
</tr>
<tr>
<td>Member3</td>
<td>1-1.0333</td>
</tr>
<tr>
<td>Member4</td>
<td>1.0111-1.0612</td>
</tr>
<tr>
<td>Member5</td>
<td>1.1231-1.222</td>
</tr>
<tr>
<td>Member6</td>
<td>1-1.0333</td>
</tr>
</tbody>
</table>

As you can see, after several rounds of Sprints, teammates gain a deeper understanding of their ability to complete tasks, thus making the estimation accuracy improved to some extent, which shows that it is effective to adjust the estimation through reliability. On the one hand, the time range of estimation is narrowed to obtain more accurate estimation results; On the other hand, it can urge members to complete positively tasks according to their own set time.

Fig. 1 shows the estimation accuracy of the team in each Sprint:

![Figure 1. Changes in estimation accuracy.](image)

The first Sprint initially establishes reliability, and the next two Sprints continuously update the personal reliability. Apparently, with the increase of the number of iterations, the estimated time adjusted by reliability gradually approaches the actual time.

B. Result Analysis

After three iterations, we collated and summarized the data of the 51 teams, then drawing a figure based on the data, which
more intuitively demonstrates that as the number of iterations increased, the estimation accuracy also increased. In the figure, one dot represents the prediction result of a team, and different colors represent different Sprints. We take the center of the first estimation time range of each team in a Sprint as a reference, calculating the prediction accuracy $k$ of the team according to (3).

$$k = \frac{d}{l}$$

(3)

Where $d$ represents the distance from the actual time to the center, and $l$ is the length of the predicted time range. When $k$ is in the range of -0.5 to 0.5, we consider the estimation to be accurate. The estimation accuracy of each Sprint is the proportion of the number of teams accurately estimated. Fig. 2 shows the estimation results of different teams.

![Figure 2. Sprint2 and Sprint3 prediction accuracy of each team.](image)

The final experiment result shows that the estimation accuracy can reach about 60% in the second Sprint and about 84% in the third Sprint. Meanwhile, the estimation error of the third Sprint is reduced compared with the second Sprint.

V. CONCLUSION

This paper mainly explores how to estimate each Sprint time through the team individual estimation in Scrum. Firstly we summarize the shortcomings of existing methods. Then we put forward the concepts of task point and ideal time and the method of applying them to Agile estimation, finally verifying the feasibility and accuracy of the method.

The advantage of the method proposed in our study is to estimation the effort of the entire team from the individual, which is an approach to estimation the effort from the actual level of individuals at a smaller granularity. This method takes into account that the estimation accuracy of individual usually will change with the progress of the project, so the reliability is proposed to modify the results, improving the estimation reliability. It implements Agile ideas and is in line with Scrum development very well. The future work will focus on how to achieve more accurate prediction for the whole project at the early stage of development and get rid of the estimation limit of current approach.

REFERENCES


ASC4AI: A Novel Automatic Service Composition Framework for AI Systems

Zhiguo Li, Qin Li*
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University
Shanghai, China

Abstract—As cloud computing becomes more prevalent in various domains, such as e-commerce, healthcare, education, etc., there is a growing demand for cloud-based applications that can perform complex tasks by integrating multiple cloud services. Despite QoS-aware automatic service composition has been extensively studied, conventional approaches face limitations in evaluating intelligent services and ensuring correctness and reliability for AI systems. To address these challenges, we propose five metrics for differentiating between intelligent and non-intelligent services and evaluating service composition solutions based on user-defined metrics constraints. We also explore ways to improve system correctness and reliability, and adapt these approaches to fit with AI systems. Building on these insights, we propose ASC4AI, a novel automatic service composition framework designed specifically for AI systems, which can automatically generate service composition solutions that meet user-defined functional and metrics constraints. Furthermore, we implement ASC4AI as a user-friendly tool that minimizes technical complexity for developers.

Index Terms—Artificial intelligent systems, Automatic service composition, Service metrics, Service composition patterns

I. INTRODUCTION

As cloud native technology advances, the use of cloud services for performing complex tasks has become increasingly popular. Additionally, to promote the diffusion and adoption of AI technologies form cloud, cloud service providers have started to offer out-of-the-box reasoning services [1]. However, service-oriented AI systems consist of a multitude of logical independent intelligent and non-intelligent services, which makes existing service composition approaches and evaluation metrics unsuitable for AI systems. Hence, it is necessary to develop a new automatic service composition framework specifically tailored to AI systems.

To our knowledge, no study has yielded a systematic solution to the problem. For example, some works propose to automatic service composition based on the service description model, but this approach not be suitable for AI systems due to lack evaluation metrics and satisfy user constraints [2]–[4]. And some other works propose to evaluate the reliability of the component-based systems [5]–[7], but user constraints extend beyond just reliability metrics.

Therefore, an intuitive idea we want to explore is, can we automatically provide service composition solutions to satisfy specific metrics constraints for AI systems? Realizing this intuition based on existing research raises two challenges: (1) Establishing distinct metrics for AI systems can help differentiate between intelligent and non-intelligent services. (2) Automatically compositing services that can meet user workflow’s functional and metrics constraints while enhancing correctness or reliability.

Regarding the first challenge, we begin by defining what is intelligent task or service in AI systems. Next, we propose five key metrics extended quality of service: correctness, reliability, throughput, response time, and GPU memory usage. Of these metrics, correctness stands out as particularly important since it provides a comprehensive measure of AI systems accuracy and can effectively differentiate between intelligent and non-intelligent services based on functionality.

To address the second challenge, we compare the semantic information of services’ input and output to determine compatibility and establish edges between matching services with associated metrics information for automatic composition. In terms of correctness and reliability, we analyze commonly used service composition patterns. Our findings suggest that fault-tolerant structures can enhance system correctness and reliability. These structures can be customized and applied to critical service nodes to effectively improve AI systems.

Based on the above explorations, we have designed a framework for automatic service composition called ASC4AI. This framework is capable of automatically generating service composition solutions that meet user-defined metrics and functional constraints, while improving the correctness or reliability of the solution. To make it more accessible, we have also developed a user-friendly tool that simplifies the technical complexity for developers or users.

The main contributions of this paper are as follows:

- We propose five metrics for AI systems: correctness, reliability, throughput, response time and GPU memory usage that can be used to differentiate between intelligent and non-intelligent services.
• We design a framework for AI systems that can automatically generate trustworthy and reliable service composition solutions and implement the framework as a user-friendly tool. The result demonstrates that our approach can be applied to AI systems.

The rest of this paper is organized as follows. In Section II, we review the related work. In Section III, we introduce the architecture of ASC4AI. In Section IV, we present the process of the proposed approach and the evaluation method. In Section V, we present the implementation of the framework and a case study. Finally, we conclude this paper in Section VI.

II. RELATED WORK
Approaches for ASC problem can be categorized into AI planning or graph-based. For instance, [8] proposes an integrated approach for automated semantic web service composition using AI planning techniques. The authors of [2] and subsequent work [3] address the automatic service composition and QoS-aware service selection by utilizing an inverted index table and a counting mechanism. However, their works focus solely on generating optimal QoS-aware compositions without taking into account user QoS requirements constraints and intelligent services.

To guarantee QoS requirements constraints of users, the QoS-aware ASC problem has attracted the attention of a lot of researchers from different fields. For example, [9] introduce a top k query mechanism to satisfy user constraints as much as possible. The work in [10] presents a novel approach based on a Harmony Search algorithm under respect global QoS constraints. In [11] the authors propose a new SOC-based approach to ensure application development which ensures the discovery, selection, and composition of the most appropriate Web services to meet the developer requirements. Unfortunately, these approaches do not consider the reliability and correctness of the AI systems.

In many cases, the effects of service faults on the business are disastrous. The fault-tolerant structure can improve the reliability and real-time capabilities of the systems [12] and the work in [13] detailed introduction of those structures. Although these works can provide guidance on how to improve systems reliability, further research is still needed on how to combine automated service composition techniques to automatically select the appropriate structure.

Overall, these approaches enable a wider search space and flexible service composition under QoS constraints, but they have limitations. Firstly, the composition patterns and QoS calculation methods are not always clear. Secondly, they do not take into account the specific requirements of AI systems.

III. ASC4AI ARCHITECTURE
The architecture of ASC4AI is illustrated in Fig. 1. The browser, representing the presentation layer, facilitates user visual interaction. The ASC4AI, serving as the logic layer, generates service composition solutions that meet the metrics constraints of the user’s workflow requirements. Finally, the database acts as the data layer and is responsible for persistently storing the data used.

The remainder of this section we describe the ASC4AI main process from user requirement to automatic service composition solution generation, the compositor will be introduced in next Section.

Users provide their workflow requirements in a flowchart format, consisting of intelligent and non-intelligent task nodes. Each node requires the description of input and output, functional requirements, and metrics constraints. For intelligent nodes, users must also specify the dataset used and evaluation method. In order to better understand the workflow requirements, we introduce the following concepts.

1) Intelligent Tasks or Services: Intelligent tasks or services are the ones that use artificial intelligence technology. They have the ability to analyze and process complex data beyond human capabilities in specific domains. However, they require extensive data training and GPU resources to function efficiently. For example, image recognition is an intelligent service, whereas adding a watermark to an image is a non-intelligent service.

2) Interaction Patterns in Workflow: The interaction between task nodes based four basic composite structure, Sequential, Parallel and Conditional from [14], shown in Fig. 2. For example, the workflow start from task node $T_1$ with output $[c, d, e]$, then the $T_2$ and $T_3$ can be executed in Parallel with input $[c, d, e]$, output $[f, g]$. Finally, the output $[h]$ is can be executed in Conditional with conditions $c_1$ and $c_2$ to get final output $[i]$.

Fig. 1: ASC4AI architecture

Fig. 2: Workflow interaction patterns

3) Metrics: To evaluate and distinguish between intelligent and non-intelligent services in AI systems, this paper proposes five metrics are as follows.
Correctness($corr$) refers to the ability of a service or component to produce accurate results. This metric is represented as a vector $corr = (e_1, e_2, \ldots, e_n)$, where $e_i$ represents the evaluation result obtained using various evaluation techniques such as confusion matrix, F1-score, ROC curve, etc. For simplifies, we use the weighted summation to calculate the correctness, which is defined as follows:

$$corr = \sum_{i=1}^{n} w_i e_i^i \quad (1)$$

where $w_i$ is the weight of the $i$-th evaluation technique, and $e_i^i$ is the normalized evaluation result of the $i$-th evaluation technique. For non-intelligent services, the correctness is always 1.

- Reliability($rel$) is the ability of a service or component to perform the required functions within a specified time interval and under specified conditions.
- Throughput($tp$) refers to the ability of the service to handle multiple requests.
- Response Time($rt$) refers to the average get response time for a single request sent to the service.
- GPU Memory($gm$) reflects the minimum required GPU memory for a service to perform optimally and often represents the service’s usage cost, with higher GPU Memory indicating higher cost.

Since the architecture cannot directly accept a flowchart, a data converter is used to translate the user’s workflow requirements into some format that can be processed by the architecture. This converter is also responsible for translating the results generated by ASC4AI back into a format that can be easily parsed by the flowchart.

The split and merge workflow is a design approach that simplifies the execution of large-scale workflows. It achieves this by dividing them into smaller sub-workflows or task nodes, processing each one separately, and then merging their results to form a single completed workflow.

After obtaining a set of service composition solutions, the user can select the most appropriate solution based on the metrics of the service composition. The user can also modify the workflow requirements and re-run the ASC4AI to obtain a new set of service composition solutions.

IV. SERVICE COMPOSITOR

Our aim is to automatically generate trustworthy and reliable service composition solutions while taking into account the constraints of user workflow requirements. The remainder of this section we describe the service composition process and evaluation method.

A. Automatic Service Composition

The automatic service composition process for one task node is shown in Fig. 3. The process includes three steps: generate a service composition graph, find a set of service sequences maybe DAGs that satisfy the constraints, and optimize reliability by replacing the key service nodes in fault-tolerant structures.

Fig. 3: Automatic service composition process

1) Generate Service Composition Graph: Generating service composition diagrams requires the use of service semantic information, and by constructing a concept tree structure, the semantic similarity between concepts can be calculated. Suppose the output of service $A$ is $O_A$ and the input of service $B$ is $I_B$. The formula for calculating the similarity between $O_A$ and $I_B$ is as follows:

$$Sim(O_A, I_B) = 1 - \sqrt{\frac{1}{2} a \cdot Dis(O_A, I_B)} \quad (2)$$

where $a = \frac{Dep(O_A)}{Dep(I_B) + Dep(O_A)}$, $Dep(X)$ is the depth of concept $X$ in the concept tree. And $Dis(O_A, I_B)$ is the distance between $O_A$ and $I_B$ in the concept tree. The distance between two concepts is defined as the length of the shortest path between them in the concept tree.

Once the similarity between two services up to a value $\theta$ (which can be provided by the user) then the link between the two services is considered to exist. After that, a service composition directed graph is constructed starting from the task node input to the output.

2) Find DAGs: After constructing the service composition directed graph, the automatic service composition problem is transformed into a directed graph path-finding problem. This paper uses the algorithm from [2] to find all the paths that satisfy the metrics constraints. But our approach differs from the original method in how we calculate these new metrics for AI systems.

In the service composition graph, it can be found the the Sequential and Parallel are deterministic, i.e., for the Sequential, the output of the previous service is the input of the next service; for the Parallel, the output of each service is the input
of the next service. So the metrics in the process of finding the optimal path are calculated in the table I.

However, the Loop and Conditional patterns in workflow interaction (see Fig. 2) involve uncertainty. For Loop, the number of loops is uncertain, and for Conditional, the probability of each condition is uncertain. So for these two patterns, we can calculate the corresponding metrics based on the component [15], [16] analysis, and a workflow is a component (task node) software with transfer probabilities (can be specified by user).

We also have some assumptions: (1) The correctness and reliability of a task node is known (determined by calculating the service composition later). (2) The control transfer between task nodes is Markovian in nature, i.e., future behavior is independent of past behavior. (3) The failure of one task node does not affect any other task node.

In this way, the calculation method for each metric can be obtained shown in Table I. In the table, \( P_{i,j} \) represents the probability of task node \( T_i \) transitioning to task node \( T_j \), while \( k \) denotes the maximum loop value collected by users.

### TABLE I: Metrics calculation method for Sequential, Parallel, and Conditional

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Sequential</th>
<th>Parallel</th>
<th>Conditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q^{corr} )</td>
<td>( \prod_{i=1}^{n} q_i^{corr} )</td>
<td>( \prod_{i=1}^{n} q_i^{corr} )</td>
<td>( P_{i,j} q_j^{corr} + P_{i,k} q_k^{corr} )</td>
</tr>
<tr>
<td>( q^{rel} )</td>
<td>( \prod_{i=1}^{n} q_i^{rel} )</td>
<td>( \prod_{i=1}^{n} q_i^{rel} )</td>
<td>( P_{i,j} q_j^{rel} + P_{i,k} q_k^{rel} )</td>
</tr>
<tr>
<td>( q^{p} )</td>
<td>( \min(q_{i1}^{p}, \ldots, q_{im}^{p}) )</td>
<td>( \min(q_{i1}^{p}, \ldots, q_{im}^{p}) )</td>
<td>( P_{i,j} q_j^{p} + P_{i,k} q_k^{p} )</td>
</tr>
<tr>
<td>( q^{t} )</td>
<td>( \sum_{i=1}^{n} q_i^{t} )</td>
<td>( \max(q_{i1}^{t}, \ldots, q_{im}^{t}) )</td>
<td>( P_{i,j} q_j^{t} + P_{i,k} q_k^{t} )</td>
</tr>
<tr>
<td>( q^{m} )</td>
<td>( \max(q_{i1}^{m}, \ldots, q_{im}^{m}) )</td>
<td>( \sum_{i=1}^{n} q_i^{m} )</td>
<td>( P_{i,j} q_j^{m} + P_{i,k} q_k^{m} )</td>
</tr>
</tbody>
</table>

3) Optimize Correctness and Reliability: AI systems used in production require accurate and reliable capabilities. However, a limitation of the previously mentioned algorithms is that if any service within the service composition solution fails or produces an error output, it can cause a system-wide failure. Since we do not have prior knowledge of a service’s internal implementation, we can only improve these capabilities of AI systems by using functionally equivalent components for fault-tolerant processing.

To more efficiently determine which service nodes should use a fault-tolerant structure, i.e., which service nodes have the greatest impact on the system, we use a method proposed in [17] as follows:

\[
F_i = \alpha_1 In_F(i) + \alpha_2 In_P(i) + (1 - \varphi) In_S(i) \tag{3}
\]

where \( In_F(i) \), \( In_P(i) \), and \( In_S(i) \) represent failure influence, fault propagation influence and self-influence of service \( i \), respectively. The user can provide the weights of the three influences, which are represented by \( \alpha_1 \), \( \alpha_2 \), and \( \varphi = \alpha_1 + \alpha_2 \).

In this way, three commonly fault-tolerant structures can be used to improve correctness or reliability capabilities are shown in Fig. 4. The following introduces the three fault-tolerant structures in detail.

- Recovery block mechanism (RB): When the main service fails, the backup services will be executed in sequence, and the recovery block mechanism will only fail when all the services have failed.
- Parallel strategy (PS): Multiple independent services are called in parallel, and the final result is determined by the return result of the first service. This strategy will only fail if all services fail to execute.
- N-version programming (NVP): Multiple independent services are called in parallel, and the final result is determined by a voting result. Typically, when \( M \) services are executing normally, N-version programming will not fail, where \( M \) is equal to \([n/2] + 1\), the number of services executing in parallel is \( n \).

It should be noted when a fault-tolerant structure is used, we assume that all services are executed and the final selection of results follows the uniform distribution. The calculation method is shown in Table II.

### TABLE II: Metrics calculation method for fault-tolerant structures

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Recovery Block</th>
<th>Parallel Strategy</th>
<th>N-version Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q^{corr} )</td>
<td>( \frac{1}{n} \sum_{i=1}^{n} q_i^{corr} )</td>
<td>( \frac{1}{n} \sum_{i=1}^{n} q_i^{corr} )</td>
<td>( 1 - \prod_{i=1}^{n} (1 - q_i^{corr}) )</td>
</tr>
<tr>
<td>( q^{rel} )</td>
<td>( 1 - \prod_{i=1}^{n} (1 - q_i^{rel}) )</td>
<td>( 1 - \prod_{i=1}^{n} (1 - q_i^{rel}) )</td>
<td>( R^* )</td>
</tr>
<tr>
<td>( q^{p} )</td>
<td>( \min(q_{i1}^{p}, \ldots, q_{im}^{p}) )</td>
<td>( \min(q_{i1}^{p}, \ldots, q_{im}^{p}) )</td>
<td>( \min(q_{i1}^{p}, \ldots, q_{im}^{p}) )</td>
</tr>
<tr>
<td>( q^{t} )</td>
<td>( \sum_{i=1}^{n} q_i^{t} )</td>
<td>( \max(q_{i1}^{t}, \ldots, q_{im}^{t}) )</td>
<td>( \max(q_{i1}^{t}, \ldots, q_{im}^{t}) )</td>
</tr>
<tr>
<td>( q^{m} )</td>
<td>( \max(q_{i1}^{m}, \ldots, q_{im}^{m}) )</td>
<td>( \sum_{i=1}^{n} q_i^{m} )</td>
<td>( \sum_{i=1}^{n} q_i^{m} )</td>
</tr>
</tbody>
</table>

\* Assuming a three-version structure, with each service’s reliability being \( q_i^{rel} \), and requiring \( M = 2 \) services to be functioning normally, the reliability of the structure is: \( R^* = q_1^{rel} q_2^{rel} q_3^{rel} + q_1^{rel} q_2^{rel} (1 - q_3^{rel}) + q_1^{rel} q_3^{rel} (1 - q_2^{rel}) + q_2^{rel} q_3^{rel} (1 - q_1^{rel}) \).

While fault-tolerant structures may decrease certain metrics such as correctness, throughput, and response time, they are generally effective at identifying a range of solutions that satisfy user constraints under normal conditions. If the user’s constraints are not met, the approach can still offer several relatively better composition solutions for the user to choose from or modify.

B. Evaluation

After the metrics of the service composition solution are calculated, the evaluation is performed, which based on the user’s constraints and the metrics of the service composition solution. These metrics are classified into positive and negative attributes. Positive attributes are characterized by higher values.
being better, such as correctness, reliability and throughput. Negative attributes are characterized by smaller values being better, such as average response time and GPU graphics memory.

Since the metrics of a service are not always within the same range, differences in such ranges will impact the final evaluation results. To address this issue, we normalize the metrics of the service to a range of $[0, 1]$ before evaluation. The resulting normalized for both positive and negative attributes can be defined as follows:

For positive, 
$$q'_i = \begin{cases} \frac{q_i - q_{i_{\text{min}}}}{q_{i_{\text{max}}} - q_{i_{\text{min}}}}, & q_{i_{\text{min}}} \neq q_{i_{\text{max}}} \\ 1, & q_{i_{\text{min}}} = q_{i_{\text{max}}} \end{cases}$$  \tag{4}

For negative, 
$$q'_i = \begin{cases} \frac{q_{i_{\text{max}}} - q_{i_{\text{min}}}}{q_{i_{\text{max}}} - q_{i_{\text{min}}}}, & q_{i_{\text{min}}} \neq q_{i_{\text{max}}} \\ 1, & q_{i_{\text{min}}} = q_{i_{\text{max}}} \end{cases}$$  \tag{5}

where $q'_i$ is the normalized metric of the service, $q_i$ is the one metric of the service, $q_{i_{\text{min}}}$ and $q_{i_{\text{max}}}$ are the minimum and maximum values of the metric, respectively.

However, users have varying preferences for different metrics. For example, some may prioritize higher throughput and lower response time. The user’s preferences can be represented by a weight vector $w = (w_1, w_2, \ldots, w_n)$, where $w_i$ is the weight of the $i$th metric. The weight vector is normalized to the range of $[0, 1]$, and the sum of the weight vector is 1. The score of the service is then calculated as follows:

$$\text{Score}_{\text{metrics}} = \sum_{i=1}^{n} w_i q'_i$$  \tag{6}

By doing this, we can obtain the score of the candidate service composition solution ($\text{Score}_{\text{composition}}$) and the constraint score of the task node ($\text{Score}_{\text{constraints}}$). The constraint score of the workflow is calculated based on the user’s workflow requirements, which are represented by a constraint vector $c = (c_1, c_2, \ldots, c_n)$, where $c_i$ is the constraint of the $i$th metric.

Once we have obtained the metrics score and constraints score, we can get the overall score ($\text{Score} = \text{Score}_{\text{composition}} - \text{Score}_{\text{constraints}}$) of the service composition solution by subtracting the two values. The overall score should be positive; otherwise, it means that the candidate service composition solution does not meet the user’s workflow requirements.

If the weighted sum method fails to meet all requirements, a full evaluation method is available. Unfortunately, it’s unlikely that all metric constraints can be met in real systems. This method involves comparing the number of metrics that a service composition solution with the given constraints and determining its domination number. The resulting value is then subtracted from 5 (the total number of metrics in this paper) to align with the weighted sum method.

V. CASE STUDY

This section demonstrates the application of the proposed approach using a traffic light recognition example from Baidu’s open-source Apollo platform in the context of advanced autonomous driving systems that contain both intelligent and non-intelligent services driven by machine learning.

The workflow example includes three tasks, as illustrated in Fig. 5. The first task detects signal lights, determines the traffic light style using an input image, and outputs the traffic light style and the cropped image. Then, a conditional node is employed to execute different tasks based on the traffic light style and probability. The remaining tasks are intelligent, but differ in the datasets used for training and styles of traffic lights applied.

![ Workflow design](image)

The generate service composition solutions shown in Fig. 6. With the aid of expert evaluation, our tool can generate service composition solutions that meet user demand constraints.

VI. CONCLUSION

In this paper, we present ASC4AI, a framework for automated service composition, which is tailored specifically for AI systems. The novelty of our framework is that we propose five metrics executed quality of service that distinguish intelligent services from non-intelligent ones. We also utilize fault-tolerant structures to improve the correctness and reliability capabilities of AI systems. Furthermore, we have developed a user-friendly tool based on the ASC4AI that has been successfully applied to intelligent manufacturing systems. Our ongoing research efforts aim to improve the algorithm efficiency and add more service composition patterns to the framework.

ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program (2020AAA0107800) and the Reliability Technology Lab of Huawei Technology Co., Ltd.
REFERENCES


A Review of Methods for Identifying Extract Method Refactoring

Omkarendra Tiwari
IIIT Allahabad, India
E-mail: omkarendra@iiita.ac.in

Vrinda Yadav
IIIT Nagpur, India
E-mail: vyadav@iiitn.ac.in

Abstract

Extract method is one of the most popular and versatile refactoring. It is primarily applied to improve the design of methods by mitigating code smells such as long method, code clone, and feature envy. In recent past, various methods for identifying extract method refactoring have been proposed. However, an established performance hierarchy among them is lacking as the proposed approaches have been evaluated on different benchmarks.

This paper evaluates the approaches in a common setting to identify various parameters and their impact on performance, with a goal to help users to identify a tool best suited to their requirement and aid researchers to make an informed decision while designing and evaluating a new approach. Existing approaches are evaluated over a common benchmark consisting of five open-source software studies with a focus on understanding the impact of evaluation settings over performance of approaches. Our experiment shows that standardization in value selection for evaluation parameters is crucial. It is observed that most of the approaches are sensitive to top-n suggestions parameter. Further, tolerance parameter used is not generalized. Our finding in performance trend measured in precision, recall, and F-measure deviate from the earlier results.

1. Introduction

Extract Method refactoring is applied to decompose a long method in order to gain benefits such as improved readability, reusability, ease of feature extension, and lower code duplicability [1, 2]. Its application can be viewed as two part process (i) identification of a task (a subset of statements in the method), and (ii) extraction of the task or extract method opportunity (EMO) as a separate method. The second part is automated and is supported in IDEs, Eclipse for example. Hence, in the recent past, researchers have proposed various techniques to automate the identification of tasks. However, little is known about practitioner’s or standard approach to this problem, for example how a set of statements is isolated as a task, what metrics are used as indicators, is there a minimum or maximum size threshold over newly extracted method, awareness and selection criteria of existing tools etc. Moreover, there exists a knowledge gap in state-of-the-art approaches.

Approaches proposed in the last decade vary in design philosophy, underlying technique, benchmarks, and evaluation criteria used. This makes it difficult for users to clearly identify best tool/method in each aspect of performance. Further, unavailability of a large benchmark makes it difficult for researchers to correctly measure the state-of-the-art. Hence, we observe a requirement for establishing a standard evaluation setup and an analysis of the recent approaches.

This study focuses over implementation (tool) and evaluation setting of the proposed approaches; learn early lessons and apply them to design benchmarks and obtain results that can be used by both researchers and practitioners. In this paper, we present an evaluation of state-of-the-art approaches over a common benchmark. Further, we discuss a set of guidelines for configuration and metrics for evaluation. Aim of the study is to bring forward standard practices which may lead to establish the merit of new research methods over existing methods. Key aspects of focus for our study are (i) benchmarks used in evaluation, and (ii) configurations and metrics used in evaluation.

Since, earlier existing approaches used different benchmarks and tool configurations for evaluation, an analysis in common setup was required to understand ’true’ performance hierarchy, this paper makes the following contributions:

- A review of state-of-the-art approaches and evaluation practices on a common benchmark
- Pluto: A new synthetic benchmark
- Guidelines for developing and evaluating new approaches

It can be noted that this is the first work aimed at studying the diverse evaluation criteria, and attempts to bring
out standard practices along with a common benchmark to help users to select a tool as per their requirements and researchers to be better informed about the state-of-the-art.

Rest of the paper is organized as follows: Section 2 lists a set of aspects of interest for researchers/developers/users when developing/selecting an extract method refactoring approach; also these aspects are the focus of this study. Evaluation of existing approaches and experimental setup is discussed in Section 3. Finally, conclusion and future work is discussed.

2. Extract Method Refactoring: Aspects of Concern

Multiple studies have been conducted to understand application and benefits of refactoring in general but little is known about extract method refactoring [1, 3]. Added with subjective nature of this refactoring, its automation becomes a challenging task. We note that human feedbacks are crucial for development of an effective and robust tool. However, there are aspects of formulation and evaluation of the refactoring approaches that are independent of expert’s feedback. So, these two aspects can be explored independently and later, findings related to the other aspect can be augmented.

```c
void FiboPrime() {
   int i, n, a, b, t;
   A. printf("Value of N:");
   B. scanf("%d", &n); // Input {A,B}
   C. a = 0;
   D. if (n == 1) {
      E. printf("Nth Term:"d",a); //Prime Checking [P-S]
      Q. else {
         F. printf("Not Prime");
         R. else // Prime Checking [P-S]
      } // Divisor Computation [M-O]
   } // Show Nth Term [L]
   M. for (i=2; i<=b/2;i++) {
      N. if (b%i == 0) {
         O. break; // Divisor Computation [M-O]
      } // Divisor Computation [M-O]
   } // Show Nth Term [L]
   L. printf("Nth Term:%d",b); // Compute Nth Term {C-K}
   J. a = b; // Compute Nth Term {C-K}
   K. b = t;
   H. for (i=3; i <=n; i++) {
      I. t = a + b;
      J. a = b; // Compute Nth Term {C-K}
      K. b = t;
   }
   } // Compute Nth Term {C-K}
   C. a = 0;
   B. scanf("%d", &n);
   A. printf("Value of N:");
   D. if (n == 1) {
      E. printf("Nth Term:"d",a); //Prime Checking [P-S]
      Q. else {
         F. printf("Not Prime");
         R. else // Prime Checking [P-S]
      } // Divisor Computation [M-O]
   } // Show Nth Term [L]
   M. for (i=2; i<=b/2;i++) {
      N. if (b%i == 0) {
         O. break; // Divisor Computation [M-O]
      } // Divisor Computation [M-O]
   } // Show Nth Term [L]
   L. printf("Nth Term:%d",b); // Compute Nth Term {C-K}
   J. a = b; // Compute Nth Term {C-K}
   K. b = t;
   H. for (i=3; i <=n; i++) {
      I. t = a + b;
      J. a = b; // Compute Nth Term {C-K}
      K. b = t;
   }
   } // Compute Nth Term {C-K}
```

Figure 1: A Program for Computing Fibonacci Prime

Now, we present aspects of extract method refactoring that are central to the study. It includes recent approaches, challenges in developing an automated solution, and evaluation setup to establish the performance hierarchy of tools.

2.1 Recent Approaches/Tools

Various refactoring approaches for Long Method code smell use techniques such as clustering [4] [5], control flow graphs [6, 7, 8] and program slicing [9], [10]. However, in this section and the paper, we restrict discussion around a few recent approaches that have been accompanied with their implementation (tool) and have evaluated their performance over open-source software studies.

One of the early extract method refactoring approach available as a tool is JDeodorant, proposed by Tsantalis and Chatzigeorgious [11, 12]. It uses complete computational slice to identify a task that can be extracted as a separate method. The computed slice may result in duplicate statements in extracted and base method. It is available as an Eclipse plug-in and have been used for performance evaluation in recent approaches.

Silva et al. [13] proposed, JExtract, a new block-based method for generating an exhaustive list of suggestions that are ranked before presenting to the user. The approach uses a web-based system MyWebMarket to identify the satisfactory configuration for the tool, which then is used in evaluating the approach via application over two open-source software studies, JUnit and JHotDraw. Authors synthetically created extract method opportunities(EMOs). The evaluation shows that the approach can achieve a high recall at low precision. Also, the results show the configuration that produces high precision lowers the recall.

Charalampidou et al. [14] proposed, SEMI, a clustering based approach for identifying extract method opportunities. The approach forms cluster of coherent statements based on the presence of a common variable, object, method name etc. This method lowers the final suggestions to the developer/user by first grouping the identified extract method opportunities and then ranking them. The approach groups similar opportunities based on overlapping statements, then finds one which offers most benefit in terms of cohesion if refactored. Similar to JExtract, it also generates an exhaustive list of suggestions.

Xu et al. [15] proposed, Gems, a machine-learning based probabilistic model for predicting extract method opportunities. Given the pair of the refactoring candidate and the method the approach extracts informations such as loop, size, invocation, type and variable access etc.

Tiwari and Joshi [16] proposed, Segmentation, a clustering based approach that aims at maximizing the precision with a manageable recall. Further, the proposed approach restricts the number of suggestions generated by implementing the policy of forming distinct clusters.

Shahidi et al. [17] proposed a method to identify, and mitigate long method code smell by application of extract method refactoring. Further, to ensure modularity the refactored code is analyzed for feature envy. Their proposed
method relies on expert’s opinion to evaluate the performance of the approach, in contrast to evaluation strategies followed by recent approaches [12, 13, 14, 16] that use synthetic benchmark.

Since, aforementioned approaches have followed varying benchmarks and evaluation criteria, a user may find it difficult to select the most suitable tool as per the requirement.

2.2 Subjectivity

Consider the Fibonacci Prime program shown in Figure 1. The program illustrates the subjective nature of this refactoring, and also exhibits the challenges in coming up with a solution. The program contains multiple subtasks packed in one method to achieve the task of Fibonacci prime computation. These subtasks are annotated on the right in the figure.

One can decompose the given method into two methods (i) statement block A-L corresponding to Fibonacci term computation, and (ii) remaining statements for Prime checking. Such a decomposition of the program would result in a Fibonacci Method with restricted resusability as it contains input-statement and display-statement within it. Though, such a decomposition is functionally correct but may not always be desirable.

Another strategy for decomposition could be to decompose statement block A-L into three parts (i) Input, (ii) Fibonacci term computation, and (iii) statement L for display. This decomposition lead to extraction of a reusable method implementing Fibonacci term computation that can be invoked with no modifications. Invocation of this method would require input parameters and returns computed Fibonacci term.

Subjectivity in decomposition adds to challenges in evaluating the performance of automated approaches. For this reason, researchers use a parameter called tolerance when matching the automatically identified task to ground truth. We discuss more on tolerance in next section.

2.3 Benchmarks

A benchmark is crucial in evaluating external behavior of the approach, its performance or usability. It also helps in gaining an insight into the improvement made by earlier approaches, and scope for future developments. Evaluation of multiple approaches over a common benchmark enables researchers to compare the approaches and rank their performance. Further, public availability of the benchmarks allows others to replicate the results and also assess the merits of their new approaches against the state-of-the-art.

Recent approaches have used three kind of benchmarks (i) Open-source software along with feedbacks from the developer, (ii) Open-source software studies, where extract method opportunities were introduced by inline refactoring, and (iii) A software from Industry. Since, this paper is focused on the aspects where direct human intervention is not present or it is restricted; the benchmark used for our study is of second kind.

2.4 Tool Configurations and Performance Metrics

Tool’s configuration is crucial in their performance report. Most of the approaches use precision, recall, and F-measure for evaluating their performance. Given that most of the approaches have shown excellent performance over either recall or precision, a change in their default or prescribed configuration would affect their ‘true’ performance. Not only the configuration but also the performance strength is crucial for a user while selecting tool for identifying refactorings. A few parameters that are used in recent approaches to tune their tools include top-n suggestions and tolerance.

The parameter top-n suggestions restricts the count of suggestions generated by a tool. This parameter directly affects both the precision and the recall of an approach. Generally, tools provide ranked suggestions. So, for example, if a tool’s first suggestion is always best match with existing refactoring opportunity in a method then considering more than one suggestions would decrease its precision. However, it can be noted that even for a method with exactly one refactoring opportunity, there are two functionality that may be extracted. For example, method shown in figure 1, a tool may identify Fibonacci or Prime as an extract method refactoring opportunity or both. Thus, it seems that value for this parameter should be greater than two.

The other parameter of interest is tolerance, which sets limit to maximum difference/mismatch between the extract method refactoring opportunity present in a method and suggested refactoring generated by the tool. If the difference between the two is within the threshold the suggestion is considered valid and counted for precision with respective tolerance limit. Finding an appropriate threshold is crucial. In literature absolute and relative threshold has been used.

3. Evaluation

This section discuss evaluation setup and analyses various configurations and their impact over performance.

3.1 Research Questions

The study is focused on quantitatively assessing the sensitivity of the approaches to experimental setup, to report findings that would be helpful for a user to make an
informed decision while selecting a tool for their source code and performance criteria.

RQ1 Does different performance parameters (precision, recall, and F-measure) have a correlation between approach?

RQ2 How does the configuration affects the performance and usage of an approach?

3.2 Experimental Setup

To assess the performance sensitivity to different experiment settings, we prepared a set of open-source software studies with extract method refactoring candidates, a set of values for tolerance threshold, downloaded and installed tools of recent approaches, and fixed precision, recall, and F-measure as parameters to measure performance.

- **Tools** In this study we use the implementation (tools) provided by the authors of the respective approaches. Further, as part of this study, of five approaches discussed in Section 2.1, we include three approaches/tools namely, JExtract, SEMI, and Segmentation. Since, this is an early phase of studying the state-of-the-art of identifying extract method refactoring, we excluded JDeodorant as it has been evaluated in multiple studies and have been outperformed on different performance measures.

We note that exclusion of the two tools (based on slicing and machine learning) lowers the diversity of the study in terms of techniques. However, included tools follow similar technique for clustering– functional-blocks-based cluster–, which adds to uniformity in evaluation. Further, included tools collectively are among the leading performers in all three performance metrics, precision, recall, and F-measure.

- **Benchmark** To analyze the aspects of the refactoring least associated or dependent on expert feedback, we use synthetic benchmark (type (ii) benchmarks discussed in Section 2.3) to evaluate the selected approaches. Moreover, synthetic benchmark is better equipped for a controlled experiment. [13] created two synthetic open-source software studies(OSS) **JUnit** and **JHotDraw**; both the OSSs were used for evaluation of multiple approaches [13, 14, 16]. We follow the same procedure to extend the set of OSSs in used in the evaluation by including **Mockito**, **EventBus**, and **JavaPoet**; collectively named as **Pluto** [18]. The OSS studies used in the paper are listed in Table 1. All five OSSs contain a total of 173 EMOs, which makes it one of the largest benchmark in terms of EMO count. Further, the table shows that individual software studies contain methods with varying range mean-size, which adds to the diversity of the benchmark.

- **Tolerance** As we discussed earlier, task of Fibonacci term computation in Figure 1 is associated with input and display subtasks. So during extraction of it, inclusion of either or both subtasks results in lower reusability as some developers may have a different source for input or they may not be willing to display result on console. In such cases, where inclusion or exclusion result in valid functional-block but its desirability is subjective, researchers use an evaluation parameter called as **tolerance**.

In literature, two methods for selecting a value for tolerance is proposed (i) absolute values in range 1-3 [16] and (ii) values relative to method size in 1%-3% [14]. Now, in case of absolute tolerance value, if the statement difference between task marked desired by expert/ground-truth and tool’s suggestion is one-statement then the suggested task is classified as match with tolerance 1. Similarly, difference of 2-statements will be classified as a match with tolerance 2, and so on. In this study, we use absolute tolerance values as for the benchmark used the mean method size and mean EMO size are 7.88 and 17.40 statements, respectively.

- **Tool Configuration** Approaches offer various options for fine tuning the analysis process. We use default configuration except for top-n suggestions option. We recorded output for top-3 and top-5 settings; these two settings have been used in literature, so we aim to analyze its impact on performance. A higher value for this setting may increase recall but would lower the precision and F-measure. Whereas, a lower value such as n=1 can be too restrictive.

3.3 Results and Discussion

This section presents and discusses results obtained by application of the considered tools over five OSS studies. Table 2 and 3 show performance of the approaches in terms of precision, recall and F-measure for top-n suggestions set.

<table>
<thead>
<tr>
<th>Name</th>
<th>OSS studies</th>
<th>#EMOs</th>
<th>Mean Method Size (LoC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pluto-1.0</td>
<td>Mockito-3.3.8</td>
<td>46</td>
<td>12.85</td>
</tr>
<tr>
<td></td>
<td>EventBus-3.2.0</td>
<td>25</td>
<td>20.48</td>
</tr>
<tr>
<td></td>
<td>JavaPoet-1.12.1</td>
<td>21</td>
<td>17.40</td>
</tr>
<tr>
<td></td>
<td>JHotDraw-5.2</td>
<td>56</td>
<td>14.98</td>
</tr>
</tbody>
</table>

Table 1: Synthetic Benchmark for Extract Method Refactoring
### Table 2: Performance for Top 5 suggestions

<table>
<thead>
<tr>
<th>Tools</th>
<th>Tolerance</th>
<th>Precision</th>
<th>Recall</th>
<th>F measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>JExtract</td>
<td>1</td>
<td>18.91</td>
<td>86.13</td>
<td>31.00</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>19.42</td>
<td>88.44</td>
<td>31.84</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>19.67</td>
<td>89.60</td>
<td>32.26</td>
</tr>
<tr>
<td>(794)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEMI</td>
<td>1</td>
<td>13.62</td>
<td>38.73</td>
<td>20.15</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>18.29</td>
<td>52.02</td>
<td>27.07</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>19.92</td>
<td>56.65</td>
<td>29.47</td>
</tr>
<tr>
<td>(492)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segmentation</td>
<td>1</td>
<td>10.88</td>
<td>12.14</td>
<td>11.48</td>
</tr>
<tr>
<td>(194)</td>
<td>2</td>
<td>23.83</td>
<td>26.59</td>
<td>25.14</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>35.23</td>
<td>39.31</td>
<td>37.15</td>
</tr>
</tbody>
</table>

### Table 3: Performance for Top 3 suggestions

<table>
<thead>
<tr>
<th>Tools (Suggestions)</th>
<th>Tolerance</th>
<th>Precision</th>
<th>Recall</th>
<th>F measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>JExtract (502)</td>
<td>1</td>
<td>28.06</td>
<td>80.92</td>
<td>41.67</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>30.06</td>
<td>86.71</td>
<td>44.64</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>30.46</td>
<td>87.86</td>
<td>45.24</td>
</tr>
<tr>
<td>SEMI (365)</td>
<td>1</td>
<td>16.44</td>
<td>34.68</td>
<td>22.30</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>22.47</td>
<td>47.40</td>
<td>30.48</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>24.93</td>
<td>52.60</td>
<td>33.83</td>
</tr>
<tr>
<td>Segmentation (192)</td>
<td>1</td>
<td>10.99</td>
<td>12.07</td>
<td>11.51</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>24.08</td>
<td>26.44</td>
<td>25.21</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>35.60</td>
<td>39.08</td>
<td>37.26</td>
</tr>
</tbody>
</table>

#### 3.3.1 Performance hierarchy (RQ1)

- **Suggestions** Segmentation is most conservative approach for generating refactoring suggestions. It provides 1.10 suggestions per EMO (top-3 suggestions). The same for JExtract and SEMI is 2.88 and 2.09, respectively. Thus, strategy used by Segmentation can be applied by existing/new approaches to restrict the suggestions generation.

- **Precision and Recall** JExtract remains top-performer for recall, whereas top performer for Precision vary between JExtract and Segmentation. For tolerance 1, JExtract is top performer in both configurations, whereas for tolerance 3, Segmentation provides best precision.

- **F-measure** A high recall with comparable precision JExtract provides high F-measure except in case of top-5 suggestion configuration at tolerance 3.

#### 3.3.2 Impact of configuration (RQ2)

- **Top-n Suggestions** We observe that a preferable value for this configuration parameter is 3 (between 3 and 5). Table 2 and 3 shows that gain in recall is 2-5% but loss in precision and F-measure is 5-10% for SEMI JExtract. JExtract exhibits most and Segmentation least sensitivity.

- **Tolerance** We note that both the absolute and relative (percentage based) criteria for tolerance are not generalizable. Absolute tolerance allows greater flexibility when EMO sizes are smaller (which is often the case) and it is too strict for large EMOs (such as 30+ lines). On the other hand, relative tolerance based on method size is not appropriate for large methods; as large methods does not guarantee relatively larger EMOs. Thus, for same sized EMO in different methods the tolerance would vary greatly (for example, tolerance for a method with 100 and 500 statements will be 1-3 and 5-15 statements respectively). For the benchmark studied, we computed median EMO and median method size for methods with 50 or more statements and found that median EMO size is 26.5 and median method size is 78.

- **Default Configuration** We observe that facilitating a user to set a default configuration settings for a session would make the process faster. For example, in this study, we needed to execute identify EMOs operation for different methods in same class. In such cases, resetting configuration for each method consumes additional time. Minimizing the number of clicks could be used as an indicator to design user configuration setting interface.

### 3.4 Threats to validity

One of the main threats to validity to our study is benchmark. It is comparatively smaller in size that is number of OSS studies included. Inclusion of additional studies would provide necessary diversity in EMOs and methods in terms of size and structure. Further, association between the individual OSSs and approaches is not evaluated, which may differ from overall performance. Finally, inclusion of additional approaches would enrich the performance hierarchy.

### 4. Conclusion

We presented an analysis of state-of-the-art approaches over an extended benchmark. The study shows that evaluation parameters such as benchmark and configuration settings are crucial in establishing true performance comparison of multiple approaches. Further, we show and discuss how values chosen for top-n and tolerance parameters can result in biased performance. Some results, such as best approach for recall or conservative suggestion generator, reaffirm the earlier findings. However, some other results, spe-
cially for tolerance, need to be reproduced for larger benchmark.

References


Towards generating contextualised fault scenarios for reliability risk analysis of cyber-physical systems

Lufeng Wei†, Zhi Li†*, Yuansong Qin†
†School of Computer Science and Engineering, Guangxi Normal University, Guilin, China
Email: weilf@stu.gxun.edu.cn, zhili@gxnu.edu.cn, wyjfqys@gmail.com

Abstract—This paper proposes an approach that combines Fault Tree Analysis with Problem Frames to present the latent reliability risks of Cyber-Physical Systems. By extracting causalities within each, this approach associates the atomic events of the Fault Tree with the shared phenomena of the Problem Diagram, in order to generate scenarios with contexts that may lead to the occurrence of faults.

Index Terms—Cyber-Physical Systems, Problem Frames, Requirements Engineering, Fault Tree Analysis

I. BACKGROUND AND MOTIVATION

The Problem Frames (PF) approach is primarily used to explain how the machine to be built will interact with the external world to meet the customer’s needs. Typically, when describing requirements using PF, forward reasoning is used - that is, triggering the modeling process from the perspective of the requirements and constructing machine behaviors by studying the properties and behaviours of the real-world domains. This modeling method is effective for Cyber-Physical Systems (CPS), but lacks the evaluation of faults caused by environmental failures. Lin et al. [1] [2] proposed the concept of Abuse Frames to introduce external attacks into the modeling process, but did not clarify the role of causalities arising from shared phenomena between domain interactions.

Fault Tree Analysis (FTA) is typically used to evaluate the failure of system components by decomposing top-level faults into intermediate events and primary events connected by logic gates in a layered manner. However, this approach often fails to represent information about the system’s requirements and other relevant factors. Additionally, according to Smith et al. [3], if the analysts neglect or do not adequately consider the contextual information of the system, they may lose the ability to determine whether combinations of system behaviors beyond the Fault Tree (FT) will lead to failures.

In the Problem Diagram (PD), shared phenomena describe interactions among domains and the machine, forming forward chains of causalities. In the FT, the generation of upper-level events is composed of lower-level events through the combination of logic gates, which can be considered to occur on backward chains of causalities. The approach proposed in this paper combines the forward and backward chains of causalities, enabling the generation of fault information with contextualised scenarios, which facilitates the detection of potential reliability problems in early analysis.

*corresponding author: zhili@gxnu.edu.cn
DOI reference number: 10.18293/SEKE2023-229

II. THE APPROACH

A. Check the Fault Tree and Create the Mapping Table

When the FT is initially constructed, its leaf nodes may be composite events [4]. To ensure that these leaf nodes have the same granularity as the shared phenomena in the PD, the first step is to check the leaf nodes in the FT and split composite events into atomic events combined by logic gates. After completing the fault tree check, and associating the FT and the PD, the fault information mapping table (FIMT) is created. For each leaf node, domain experts should look for the corresponding shared phenomena in the PD, thus shared phenomena that can be matched to atomic events are set as associated elements in the FIMT.

B. Minimum Cut Sets Extraction

The top event in the FT can be generated by a combination of different leaf nodes. In this paper, a top down method and Boolean algebra operation rules are used to split the top event into an event set composed of leaf nodes, and then reduce the redundant items in the event set to obtain minimum cut sets. Minimum cut sets will be used for further analysis in step D.

C. Causal Chain Set Extraction

The causal attributes of the domain in the PD are part of causal chains. To obtain causal chains in the PD, the following cases: \( a \rightarrow (b \land c) \), \( a \rightarrow (b \lor c) \), and \( (a \lor b) \rightarrow c \) should be split into one-to-one causalities, where \( a \), \( b \), and \( c \) are shared phenomena in domains. This paper regards the causal transformation of shared phenomena in the PD as a directed acyclic graph and represents it as an adjacency matrix. The DFS algorithm is used to obtain a set of causal chains that start from the initial shared phenomenon and end at the terminal shared phenomenon. To obtain a set of causal chains that include all shared phenomena, this paper also treats each shared phenomenon that does not participate in causal transformation as a causal chain containing only one element.

D. Fault Scenario Generation

If all events in a minimal cut set occur, the top event of the FT is guaranteed to occur. For each atomic event in a given cut set, this paper search for the corresponding shared phenomenon in the causal chain set of the PD based on the FIMT. If a corresponding shared phenomenon can be found in the FIMT for each atomic event, the cut set can be triggered by the known domain in the PD. Treating
the mapping information in the FIMT as connecting nodes, with the predecessors being the shared phenomena that can be associated with the causal chain set and the successors being the events in the minimal cut set of the fault tree, a contextualised fault scenario starting from the initial shared phenomenon and ending at the top event of the FT can be obtained. If the atomic events included in a fault scenario can form a minimal cut set and the cut set can be triggered by the known domains in the PD, we can draw the conclusion that the ending of this scenario is reachable, and this scenario is suitable to describe the system’s fault trace.

III. CASE STUDIES

This section presents an example of railway barrier control system. In this example, a sensor is set to detect the distance between train and crossing. The barrier gates are required to keep open before and after the train passes through the crossing, allowing cars to pass the crossing normally; when the train is about to arrive, both sides of the barrier gates need to be closed to prevent cars from entering the crossing. The PD, FT, full version of causal chain set and minimal cut sets for this example are shown at GitHub¹, the FT of this example indicates that a traffic accident will occur when a car enters the crossing, the barrier gates closed, and the train enters the crossing.

Matching the shared phenomena in the PD to the relevant FT leaf nodes, we can get the FIMT partly shown at Table I, which means event X₁, X₃ and X₅ in the FT can be found in the PD.

Traversing the causal transformations in the PD, the following parts of causal chains can be extracted:

\[
\text{Train\textunderscore approach} \rightarrow \text{Train\textunderscore approaching} \rightarrow \text{Fence\textunderscore off} \rightarrow \text{Fence\textunderscore closed} \quad (1)
\]

\[
\text{Train\textunderscore in} \quad (2)
\]

\[
\text{Car\textunderscore in} \quad (3)
\]

By comparing with the FIMT, it can be found that for the minimum cut set \{X₁, X₂, X₃\}, the causal chains extracted from the PD contain shared phenomena that make all three atomic events hold. Therefore, the following fault scenario with contextual information can be obtained.

\[
\text{Causal Chain (3)} \rightarrow \text{E₁} \rightarrow X₁ \\downarrow
\]

\[
\text{Causal Chain (1)} \rightarrow \text{E₅} \rightarrow X₅ \rightarrow \text{Cutset} \rightarrow \text{Top event}
\]

\[
\text{Causal Chain (2)} \rightarrow \text{E₃} \rightarrow X₃ \\uparrow
\]

This scenario illustrates that the combination of the car’s entrance into the crossing, the train’s approach from distance and entrance into the crossing can lead to a fault, even though all three behaviors are normal in the system. Therefore, compared to the causal transformation information obtained solely from the PD (predecessors of connecting nodes E₁-E₃), this method can reveal whether the combination of normal behaviors within the system will cause a fault.

In addition, this scenario also demonstrates a fact that \text{Fence\textunderscore closed} is caused by \text{Train\textunderscore approach}, which could not be found in the FT. Therefore, compared to purely considering the combination of atomic events in the FT (successors of connecting nodes E₁-E₃), this approach can identify the contextual information when the top event occurs. By generating fault scenarios from two perspectives, analysis personnel can discover defects in software architecture in the early stages of a project and avoid introducing potential reliability issues into the development phase.

IV. CONCLUSION

This paper combines the Problem Frames with the Fault Tree Analysis to overcome the shortcomings of both methods and obtain a more complete description of fault scenarios by leveraging the advantages of both, which is crucial for reducing latent errors in the requirements analysis phase. In future work, we will refine the extraction of causalities from the Problem Diagram and attempt to refine the approach used to describe the domain properties and extract causal information from outside the machine, in order to obtain more reasonable causal contextual scenarios.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (61862009), Guangxi “Bagui Scholar” Teams for Innovation and Research.

REFERENCES


¹https://github.com/Wei–GXNU/Railway–Example

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Type</th>
<th>Domain</th>
<th>Shared Phenomenon</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>X1</td>
<td>Detectable</td>
<td>Car</td>
<td>Car_in</td>
</tr>
<tr>
<td>E3</td>
<td>X3</td>
<td>Detectable</td>
<td>Train</td>
<td>Train_in</td>
</tr>
<tr>
<td>E5</td>
<td>X5</td>
<td>Detectable</td>
<td>Fence</td>
<td>Fence_closed</td>
</tr>
</tbody>
</table>
Session SEAI: Software Engineering and AI
CCGRA: Smart Contract Code Comment Generation with Retrieval-enhanced Approach

Zhenhua Zhang  
College of Intelligence and Computing  
Tianjin University  
Tianjin, China  
zzhbible@tju.edu.cn

Shizhan Chen  
College of Intelligence and Computing  
Tianjin University  
Tianjin, China  
shizhan@tju.edu.cn

Guodong Fan  
College of Intelligence and Computing  
Tianjin University  
Tianjin, China  
guodongfan@tju.edu.cn

Guang Yang  
The College of Computer Science and Technology  
Nanjing University of Aeronautics and Astronautics  
Nanjing, China  
novelyg@outlook.com

Zhiyong Feng  
College of Intelligence and Computing  
Tianjin University  
Tianjin, China  
zyfeng@tju.edu.cn

Abstract—Smart contracts are self-executing programs on the blockchain that are critical to a range of industries, including finance, supply chain management, and healthcare. However, comprehending smart contracts can be challenging due to a lack of effective comments in most user-defined code. To address this challenge, we propose a novel retrieval-enhanced approach CCGRA that leverages retrieval knowledge to generate high-quality comments for Solidity language code. Our approach carefully eliminates duplicated data and template data in the widely-used smart contract dataset to ensure a high-quality corpus. Extensive experiments and comprehensive analysis demonstrate the effectiveness applicability of our approach after being compared with eight state-of-the-art baselines. Finally, we conduct a human study and find the comment quality generated by our approach is better than baselines in terms of similarity, naturalness, and informativeness.

Index Terms—Code Comment Generation, Smart Contract, CodeT5, Information Retrieval

I. INTRODUCTION

Smart contracts are self-executing programs that reside on the blockchain. They comprise of template code and user-defined code. Template code provide basic functionalities such as asset transfer, voting, or escrow, while user-defined code tailors the smart contract to specific needs, including business logic and rules. However, user-defined code is often not commented, making it challenging for traders to read, especially in high-stakes transactions [1].

Adding precise comments to user-defined code is critical for contract code comprehensibility and trust-building. However, the quality of the dataset used for training comment generation models can significantly impact their ability to accurately capture the nuances of user-defined code [2]. Our investigation highlights that the Solihty datasets [3], which widely used for smart contract code comment generation tasks, contains a significant amount of template code data that can skew the model’s learning towards the template code. This, in turn, compromises the model’s ability to generate high-quality comments for user-defined code. To address this challenge, we propose a solution that involves removing duplicate template code comments, which can lead to more effective and accurate comment generation for user-defined code.

In this paper, we propose a novel retrieval-enhanced approach CCRA (Code Comment Generation with Retrieval-enhanced Approach). Our approach leverages the advantages of pre-trained language models and retrieval techniques to generate reliable and high-quality comments for smart contract. To achieve this objective, we utilize the widely-used smart contract dataset [3], carefully eliminating duplicated data and template data to ensure a high-quality training set. In the end, the dataset we used contains 29,720 (code, comment) pairs, and the experimental results indicate that it outperforms eight state-of-the-art baselines. Our approach showcases the potential of combining pre-trained language models and retrieval techniques to improve the quality of generated comment in the smart contract.

The main contributions can be summarized as follows:

- We address the challenges associated with current dataset used for smart contract code comment generation by eliminating duplicated data and template data.
- We conduct a comprehensive empirical study and a human evaluation to evaluate the performance of various baselines and demonstrate that our proposed approach, CCRA, achieves state-of-the-art results.
- We share our corpus and scripts on our project homepage [4] to promote the replication of our research.

II. RELATED WORK

A. Code Comment Generation

1) Information retrieval-based approaches: In the early study phase, the researchers aimed to retrieve similar code

DOI reference number: 10.18293/SEKE2023-090

https://github.com/ZZHbible/CCGRA
from the software repository to improve the quality of code comments. Haiduc et al. [4, 5] considered two information retrieval models VSM and LSI. Rodeghero et al. [6] utilized the eye-tracking technique to identify the statements and keywords focused on by the developers.

2) Deep learning-based approaches: Recently, most of the previous studies followed deep learning-based approaches and achieved promising results. For example, Hu et al. [7] proposed the approach DeepCom by analyzing abstract syntax trees (ASTs). Later, Hu et al. [8] further proposed the improved approach Hybrid-DeepCom. Ye et al. [9] and Yang et al. [10] exploited the probabilistic correlation between the code summarization and code generation task via dual learning.

Ahmad et al. [11] used the Transformer model to generate code comments. The Transformer model is a kind of Seq2Seq model based on multi-head self-attention, which could effectively capture long-range dependencies. Then the first pre-trained model CodeBert on the source code was proposed by Feng et al. [12]. Later, Ahmad et al. [13] proposed another pre-training model PLBART for the source code. Recently, Wang et al. [14] proposed the pre-training model CodeT5.

3) Retrieval-enhanced approaches: Except for the approaches based on information retrieval or deep learning, recent studies also proposed approaches, which could fuse information retrieval and deep learning. Zhang et al. [15] were the first to take advantage of both information retrieval and deep learning approaches and proposed the approach Rencos. Wei et al. [16] used the comment of the similar code snippet as exemplars and proposed the approach Re2Com.

B. Code Intelligence in Smart Contract

1) Vulnerability detection and repair: Code intelligence in smart contracts is its ability to detect and prevent security vulnerabilities. Code intelligence can analyze the code and identify potential security flaws, such as re-entrancy attacks, and recommend appropriate measures to mitigate them. Tsankov et al. [17] release static analysis tools named Securify used to analyze code vulnerabilities in smart contracts. Zhang et al. [18] proposed a system for automatically repairing code.

2) Smart contract code comment generation: The comment generation of smart contracts can effectively help traders understand the contract content. Shi et al. [19] propose an automated translation approach based on AST, and leveraged reinforcement learning to train a syntax synthesizer to generate comprehensible comments. Hu et al. [20] exploited the Transformer and Pointer mechanism to learn the representation of source code and generates natural language descriptions.

In this study, our main focus is on combining the retrieval augmentation method with a pre-training model, leveraging the existing code library and the pre-training model generalization ability for generating comments for smart contract code.

III. APPROACH

In this section, we introduce the framework of CCGRA, which is illustrated in Fig. 1. Overall, CCGRA consists of two modules: (a) Retrieval Module. This module retrieves the most similar candidate code from the corpus to the given code. The comment associated with the retrieved code is then used as the basis for generating comment on the given code. (b) Generation Module. This module combines the retrieved comment with the given code, adds a prompt, and learns to generate the comment using the prior knowledge of a pre-trained language model.

A. Retrieval Module

Suppose we index the corpus into a list of key-value pairs, i.e. \( Z = \{(x_i, y_i)\} \), where \( x \) means the code and \( y \) means the comment. Then, given the input code \( x \), the retrieval module \( \Theta \) matches it with all code and returns the most similar code together with its comment:

\[
\Theta(x \mid Z) = \{(x'_i, y'_i)\}
\]  

Fig. 1. The framework of CCGRA

In this work, we build the retrieval engine based on the CCGIR [21]. Therefore, for the given input code \( x \), we define the input code sequence \( \{x_i\}_{i=1}^M \), where \( M \) is the length of the code sequence. Then we encode the sequence via CodeBert, extract the hidden states to get the semantic vector \( X \in \mathbb{R}^d \), which \( D \) denotes the hidden dimension of CodeBert. Then we further perform a linear transformation of \( X \) via BERT-whitening get \( \tilde{X} \in \mathbb{R}^d \), which can reduce its dimension from \( D \) to \( d \). Thus for the target code snippet \( x \) and the code snippet \( x_i \) in the corpus, we can get the semantic vectors \( X \) and \( \tilde{X}_i \). Then we can calculate their dot product score as their semantic similarity to select the top - \( k \) most similar code snippets as the candidates from the corpus \( Z \).

To better combine syntactic and lexical knowledge of the source code, we separately utilize syntactic-level similarity and lexical-level similarity to find the most similar code \( x' \). Since the computational cost of calculating the similarities based on tree matching algorithms is high, we generate its corresponding AST sequence for each code snippet and then calculate the syntactical-level similarity via the edit distance.
Since keywords occur more frequently than other tokens in the source code, these duplicated keywords may have a negative impact. We calculate the lexical-level similarity based on the set structure and Jaccard score.

Finally, we use $\text{mixed\_score}$ to retrieve the most similar code. For the code snippets $x_1$ and $x_2$, $\text{mixed\_score}$ can be calculated as follows.

$$\text{mixed\_score}(x_1, x_2) = \lambda \times \text{lexical\_similarity}(x_1, x_2) + (1 - \lambda) \times \text{syntactic\_similarity}(x_1, x_2)$$ (2)

where $\lambda$ is a hyper-parameter for knowledge fusion, which can control the ratio of the lexical-level similarity and the syntactical-level similarity.

B. Generation Module

For the given input code $x$, we use the retrieval engine to find the the most similar code $x'$ together with its comment $y'$. As retrieval from a large corpus is computationally costly, we propose to retrieve from the labeled training data. In other words, we directly adopt the training data $T = \{ (x_1, y_1), \ldots, (x_N, y_N) \}$ as the indexed corpus $Z$, where $x_i$ is the input code and $y_i$ is the ground-truth comment. Note that during training, as the input code $x$ is already indexed, we filter it from the retrieval results to avoid data leakage.

Inspired by Instruct-GPT [22], constructing instructions for downstream tasks can stimulate the potential of pre-trained models. For our task, we design the template function $f$ by appending task-specific instructions as follows.

$$f(x, y') = \text{"summarize Solidity : } y' \oplus x\text{"}$$ (3)

Here we use $f(x, y')$ to denote the input of the pre-trained model, and we use CodeT5 [14] as the backbone model for our task. The encoder inputs $f(x, y')$ and outputs the hidden representation $h = \text{Enc}(f(x, y'))$. Then the decoder iterates on the previously generated token $y < j$ via self-attention, and then predicts the probability of the next text token $P_{\phi}(y_j | y_{<j}, x) = \text{Dec}(y_{<j}, h)$. We train our model $\Phi$ by minimizing the negative log-likelihood of the target text tokens $y$ for a given input $f(x, y')$. The formula can be defined as follows

$$\mathcal{L}_\Phi = - \sum_{j=1}^{y} \log P_{\phi}(y_j | y_{<j}, f(x, y'))$$ (4)

IV. EXPERIMENTAL SETUP

In our empirical study, we aim to answer the following three research questions (RQs).

RQ1: How effective is our CCGRA compared to the baseline models in terms of automatic performance measures?

RQ2: How do various retrieval methods affect the retrieval-augmented pre-training model?

RQ3: How effective is our CCGRA at generating higher-quality comments in terms of human evaluation?

A. Experimental Subjects

In our empirical study, we employed a carefully selected corpus of smart contracts sourced from Etherscan.io and provided by Zhuang et al. [3] and Yang et al. [21]. To ensure a high-quality dataset, we carefully eliminated duplicate data and template data. Our final dataset consisted of 29,720 pairs of data, which we split into training, validation, and testing sets in an 8:1:1 ratio. We also computed the average number of tokens in both code and comments, providing detailed statistics in Table I.

<table>
<thead>
<tr>
<th>Type</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>23,776</td>
<td>2,972</td>
<td>2,972</td>
</tr>
<tr>
<td>Avg. tokens in code</td>
<td>80.54</td>
<td>80.13</td>
<td>82.27</td>
</tr>
<tr>
<td>Avg. tokens in comment</td>
<td>12.05</td>
<td>11.97</td>
<td>12.10</td>
</tr>
</tbody>
</table>

B. Performance Evaluation Measures

In our experimental study, we use three performance evaluation measures (i.e., $\text{BLEU}$ [23], $\text{METEOR}$ [24], and $\text{ROUGE-L}$ [25]) from the source code summarization domain to automatically evaluate the quality of the generated comments. Moreover, these performance measures have also been widely used in previous studies for source code summarization [26], which can alleviate the construct threats of our empirical study.

To avoid the result difference due to different performance measure implementation versions [27], we utilize the nlg-eval package [3], which can ensure the implementation correctness of these performance measures and guarantee a fair comparison.

C. Baselines

To show the competitiveness of our proposed approach CCGRA, we evaluate our proposed approach against eight state-of-the-art source code summarization baselines. Specifically, we classify these baselines into three groups. The first group is information retrieval approaches, including BM25 [28], NNGen [29], and CCGIR [21]. The second group is deep learning approaches, including CodeBert [12], UniXcoder [13], and CodeT5 [14]. The last group is hybrid approaches, including Rencos [15] and BashExplainer [30].

D. Experimental Settings

In our empirical study, we use the packages Faiss [3] and Transformers [6] to implement our proposed approach CCGRA. The hyper-parameters and their values in our empirical study are summarized in Table II.

All the experiments run on a computer with an Intel(R) Xeon(R) Silver 4210 CPU and a GeForce RTX3090 GPU with 24 GB memory. The running OS platform is Ubuntu operation system.

TABLE II
HYPER-PARAMETERS AND THEIR VALUES IN OUR EMPIRICAL STUDY

<table>
<thead>
<tr>
<th>Module</th>
<th>Hyper-parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieval</td>
<td>Number of layers</td>
<td>12</td>
</tr>
<tr>
<td>Retrieval</td>
<td>Number of multi-attention-heads</td>
<td>12</td>
</tr>
<tr>
<td>Retrieval</td>
<td>Dimension $d_{\text{after BERT-whitening}}$</td>
<td>256</td>
</tr>
<tr>
<td>Retrieval</td>
<td>Coefficient $\alpha$ of mixed score</td>
<td>0.7</td>
</tr>
<tr>
<td>Generation</td>
<td>Model size $d_{\text{model}}$</td>
<td>768</td>
</tr>
</tbody>
</table>

V. EXPERIMENTAL RESULTS

A. RQ1: How effective is our CCGRA compared to the baseline models in terms of automatic performance measures?

In this RQ, we want to investigate how effective our approach is and how much performance improvement our approach can achieve over the baselines.

TABLE III
COMPARISON BETWEEN BASELINES AND CCGRA

<table>
<thead>
<tr>
<th>Model Name</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25</td>
<td>20.18</td>
<td>16.98</td>
<td>16.81</td>
<td>37.08</td>
</tr>
<tr>
<td>NNGen</td>
<td>21.33</td>
<td>18.17</td>
<td>16.83</td>
<td>37.70</td>
</tr>
<tr>
<td>CCGIR</td>
<td>22.12</td>
<td>18.96</td>
<td>17.33</td>
<td>38.12</td>
</tr>
<tr>
<td>CodeBert</td>
<td>19.34</td>
<td>16.61</td>
<td>17.38</td>
<td>42.15</td>
</tr>
<tr>
<td>UniXcoder</td>
<td>19.49</td>
<td>16.56</td>
<td>17.03</td>
<td>40.46</td>
</tr>
<tr>
<td>CodeT5</td>
<td>23.16</td>
<td>20.48</td>
<td>19.82</td>
<td>45.86</td>
</tr>
<tr>
<td>Rencos</td>
<td>17.35</td>
<td>14.53</td>
<td>14.95</td>
<td>38.45</td>
</tr>
<tr>
<td>BashExplainer</td>
<td>21.09</td>
<td>18.62</td>
<td>18.22</td>
<td>42.62</td>
</tr>
<tr>
<td>CCGRA</td>
<td>25.55</td>
<td>22.20</td>
<td>20.84</td>
<td>46.32</td>
</tr>
</tbody>
</table>

The results are shown in Table III where the best results are in bold fonts. We can find that we proposed CCGRA achieve the best results among the eight baselines. By comparing the average growth rate of four metrics with the baseline models, our proposed CCGRA outperforms them by a substantial margin of 26.58%, 22.17%, 18.58%, 23.88%, 25.49%, 6.21%, 30.12%, and 9.57% for BM25, NNGen, CCGIR, CodeBert, UniXcoder, CodeT5, and BashExplainer, respectively. And the results indicate that retrieval-based methods outperform the CodeBert and UniXcoder models in terms of BLEU scores, suggesting that the comments generated by retrieval-based methods are effective in maintaining language consistency. However, in terms of ROUGE-L scores, pre-trained models perform better than retrieval-based methods, indicating that the comments generated by pre-trained models match the reference comments more closely in terms of semantic similarity. Moreover, it can be observed that the transformer with an encoder-decoder structure has better performance in generating Solidity comment (which does not appear in the pre-training data) compared to models with a single encoder structure.

To further demonstrate the effectiveness of CCGRA, we conduct qualitative analysis and two examples of generated comments are listed in Fig. 2. We denote comments written by humans in bold black font, and highlight comparisons in light blue and red. From Fig. 2(a), we find that through comparing the light blue and red comments that CCGIR and BashExplainer generated “activities” and “contributors” respectively, which is not correct. Meanwhile, CodeT5 generated “partner account” which is not as accurate as human-written comments. Only CCGRA is more accurate in identifying the objects in generating comments compared to the approaches of retrieval and pre-trained models. From Fig. 2(b), we find that when generating long text comments, the comments generated by retrieval-based approaches are not accurate enough, and those generated by pre-trained models are relatively short and lacked comprehensiveness. However, CCGRA can effectively overcome the shortcomings of these approaches and generate comments that are most in line with human-written comments. CCGRA combines the advantages of retrieval-based approaches by retrieving similar comment and constructing prompts to reduce semantic bias in generating comments on unseen datasets during pretraining, achieving optimal results.

Summary for RQ1: CCGRA can achieve better performance than eight state-of-the-art baselines in automatic evaluation.

B. RQ2: How do various retrieval methods affect the retrieval-augmented pre-training model?

To show the relative importance of retrieval-based approach in CCGRA, we perform a series of ablation studies over the key modules.

TABLE IV
COMPARISON BETWEEN DIFFERENT BASELINES

<table>
<thead>
<tr>
<th>Model Name</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>UniXcoder</td>
<td>19.49</td>
<td>16.56</td>
<td>17.03</td>
<td>40.46</td>
</tr>
<tr>
<td>with BM25</td>
<td>21.36</td>
<td>18.23</td>
<td>18.82</td>
<td>44.32</td>
</tr>
<tr>
<td>with NNGen</td>
<td>21.56</td>
<td>18.56</td>
<td>18.34</td>
<td>43.89</td>
</tr>
<tr>
<td>with CCGIR</td>
<td>23.16</td>
<td>20.48</td>
<td>19.82</td>
<td>45.86</td>
</tr>
<tr>
<td>CodeT5</td>
<td>23.16</td>
<td>20.48</td>
<td>19.82</td>
<td>45.86</td>
</tr>
<tr>
<td>with BM25</td>
<td>24.42</td>
<td>21.01</td>
<td>20.52</td>
<td>46.02</td>
</tr>
<tr>
<td>with NNGen</td>
<td>23.86</td>
<td>20.72</td>
<td>20.05</td>
<td>45.56</td>
</tr>
<tr>
<td>with CCGIR</td>
<td>25.55</td>
<td>22.20</td>
<td>20.84</td>
<td>46.32</td>
</tr>
</tbody>
</table>
Table IV shows the performance comparison of several retrieval-based approaches combined with pre-trained models. We observe that an effective retrieval approach can promote the model to learn the representation of the code, thereby guiding the generation of appropriate comments by the model. Meanwhile, the results show that BM25, NNGen, and CCGIR can improve UniXcoder and CodeT5 by 10.08%, 12.08%, 23.67%, and 2.57%, 1.17%, 8.40%, respectively, in terms of the BLEU-4 score. Furthermore, our results show that an improved retrieval ability is positively correlated with the model’s learning ability. We find that the performance of the retrieval module directly impacted the model’s overall performance, which can highlight the positive impact of retrieval augment.

Summary for RQ2: The incorporation of a retrieval module into the overall model has a significant impact on its performance. Specifically, the performance of the retrieval module is positively correlated with the overall model’s effectiveness.

C. RQ3: How effective is our CCGRA at generating higher-quality comments in terms of human evaluation?

Although automatic performance metrics can evaluate the gap between the generated comments and reference comments written by humans, these performance measures may not truly reflect the semantic similarity between different comments. To verify the effectiveness of our proposed approach CCGRA, we further conducted a human study. In our human study, we only compare CCGRA with CodeT5, which can achieve the best performance in the all baselines. We follow the methodology used by Wei et al. and Yang et al. to conduct the human evaluation from three aspects:

- **Similarity** evaluates the semantic similarity between the generated comments and the reference comments.
- **Naturalness** evaluates the fluency of the generated comments.
- **Informativeness** evaluates the amount of content transferred from the code to the generated comments.

We invite five master students, who have 1~3 years of smart contract experience and have good English reading ability. Due to the high cost of manually analyzing all these samples in the testing set, we use a commonly-used sampling method to select the minimum random samples. The number of selected samples can be determined via the following formula:

\[ MIN = \frac{n_0}{1 + \frac{n_0 - 1}{\text{size}}} \]  \hspace{1cm} (5)

where \( n_0 \) is related to the confidence level and the error margin \( n_0 = Z^2 \times 0.25 \). Here \( Z \) is the confidence level score and \( e \) is the error margin. \( \text{size} \) is the size of the testing set. In this RQ, we select \( MIN \) examples with the error margin \( e = 0.05 \) at 95% confidence level. Specifically, we randomly selected 340 samples from the corpus.

For each code snippet, we generate a questionnaire for each participant. Each participant is asked to score each comment in terms of similarity, naturalness, and informativeness. We observe that an effective retrieval approach can promote the generation of appropriate comments by the model. By leveraging retrieval techniques and pre-trained language models, CCGRA is able to produce reliable and informative comments that improve the comprehensibility and trust-building of smart contract code. We have demonstrated the effectiveness of our approach through extensive experiments and comprehensive analysis.

Summary for RQ3: Our human study shows that CCGRA can generate higher quality comments in terms of similarity, naturalness, and informativeness.

**VI. Threats to Validity**

**Internal threats.** The internal threat is the potential defects in the implementation of our proposed approach and baselines. To alleviate this threat, we first check the code carefully and re-implement baselines according to the original studies.

**External threats.** The external threat is the choice of corpora. To alleviate this threat, we select the popular corpora, which have been widely used in previous studies on smart contract code summarization.

**Construct threats.** This threat relates to the suitability of our selected performance measures. To alleviate this threat, we consider the widely used performance measures and also conduct a human study to verify the effectiveness of our proposed approach.

**VII. Conclusion**

In this study, we propose a novel retrieval-enhanced approach CCGRA for generating high-quality comments for user-defined code in smart contracts. By leveraging retrieval techniques and pre-trained language models, CCGRA is able to produce reliable and informative comments that improve the comprehensibility and trust-building of smart contract code. We have demonstrated the effectiveness of our approach through extensive experiments and comprehensive analysis.
addition, a human study was conducted to show that the quality of comments generated by CCGRA outperforms baselines in terms of similarity, naturalness, and informativeness.

In the future, we aim to further improve the performance of CCGRA by exploring advanced code representation methods. Additionally, we plan to expand our dataset by mining more high-quality data of smart contracts, which will facilitate the practical application of our research in various industries, such as finance and healthcare.

ACKNOWLEDGMENT
This work is supported by the National Natural Science Key Foundation of China grant No.62032016 and No.61832014.

REFERENCES
Detecting Design Patterns As Described By An Explicit Specification

Ed van Doorn  
*Faculteit Bètawetenschappen*  
Open University of the Netherlands  
Heerlen, the Netherlands  
ed.vandoorn@ou.nl

Sylvia Stuurman  
*Faculteit Bètawetenschappen*  
Open University of the Netherlands  
Heerlen, the Netherlands  
sylvia.stuurman@ou.nl

Abstract—Teaching design patterns in higher education enhances coding skills and improves software quality. However, manually correcting their work and detecting whether the students have correctly implemented the design patterns is time-consuming and would benefit from automated tools. There exist tools that can automatically detect design patterns in code. However, these tools rarely provide an explicit specification of what each pattern should look like. As a consequence, these tools are not usable in an educational environment. When using a tool to detect design patterns in an educational environment, there are several important considerations to keep in mind. Firstly, the tool should use an explicit specification of each design pattern. Secondly, as a teacher, you should be able to adjust the specifications. Third, the tool should be accurate, meaning it should produce few to no false positives or false negatives when identifying patterns in the students’ code. Finally, the tool should do this computationally efficiently. In this article, we present a new tool for detecting design patterns in code. Our tool is capable of recognizing 20 of the most commonly used design patterns, including those described by the ‘Gang of Four’.

*Index Terms*—Design Patterns, Detection, Specification

I. INTRODUCTION

Design patterns have shown a positive impact on the maintainability of software [14]. As a result, teaching design patterns has become an important topic in software engineering education at the university level.

Design patterns were originally described in the book written by ‘the Gang of Four’ (GoF) [5]. However, the book primarily uses natural language, class diagrams, and examples in C++ source code to describe the patterns, without providing a clear and unambiguous specification. This can make it challenging for teachers to assess assignments that require the use of design patterns, and can lead to time-consuming discussions with students over the correct implementation of these patterns. As a result, there is a need for tools that can help automate the process of detecting design patterns in code, and provide a more objective way of evaluating students’ work. Although automated tools for detecting design patterns exist, many of them lack transparency in their detection methodology. In an educational environment, it is essential for teachers to have access to tools that can accurately detect design patterns used in student code while being customizable to only identify specific patterns. Additionally, the tool should have a low rate of false positives and false negatives to ensure accurate pattern detection. Finally, the tool should be computationally efficient to enable teachers to efficiently evaluate large amounts of code.

To address this issue, we have developed a more precise specification of the ‘GoF’ patterns [4], which can be customized to suit the preferences of individual teachers. In this article, we will demonstrate the accuracy of our tool, which is based on this specification. We assess our tool using a test set that includes all the GoF design patterns, and Java sources that were found ‘in the wild’. By using this approach, we can ensure that our tool accurately detects design patterns in a variety of contexts and scenarios.

The contributions of this article are:
- An algorithm based on explicit specifications that can detect 20 GoF design patterns.
- A reusable test set for the GoF design patterns.
- An assessment where we demonstrate the efficiency and accuracy of our approach.

The remainder of this paper is organized as follows. Section II describes related work. In Section III, we outline the specifications we use and describe our tool in detail. In Section IV, we present the results of our tool’s verification and validation. Finally, Section V contains our conclusions, a discussion of our work, and suggestions for future research.

II. RELATED WORK

Hadis Yarahmadi et al. have written a comprehensive systematic review of design pattern detection methods that are published in 112 articles between 2008 - 2019 [13].

None of the authors describe the specifications of design patterns that their software can detect. Nevertheless, they list the design patterns that can be detected by their software.

The detection methods can be divided into two groups: *exact* and *inexact*. A detection method is *inexact* if, after detecting a part of the design pattern, the software can report that the design pattern has been recognized.

Inexact detection methods will produce more false positives than exact detection methods.

In an educational environment, we want only to detect completely implemented design patterns. So, we focus on
an exact detection method, using an explicit specification of design patterns.

We only found a few works after 2019 complementing the review from Yarahmadi et al. We will describe them next.

For several years, machine learning has been used to detect design patterns. An example is a recent study by [8], in which fifteen features of code are defined, such as class names, interfaces, methods, parameters, and a number of variables in a method. These features are used to construct word vectors [7] of n-grams, which are groups of n consecutive words.

Nazar’s approach is based on an ensemble [10] of randomized decision trees, which classifies a design pattern. These ensembles are combined with a supervised learning algorithm.

To train a supervised neural network, 1300 Java files from Github were selected. Every Java file either contained one of twelve design patterns or did not contain any design pattern. As a benchmark, 1039 Java files from P-Markt were used. The benchmark was used for calculating the precision and recall.

This resulted in an average precision of 80% and an average recall of 79%.

III. OUR APPROACH TO DESIGN PATTERN DETECTION

A. Basis of our approach

Our approach builds upon two key foundations. Firstly, it improves upon an earlier detection tool that we developed [2]. Secondly, it is based on our construction of explicit specifications for design patterns [4].

The earlier detection tool utilized exact subgraph matching and relied solely on the names of the participating classes and their relationships. Consequently, it could only detect design patterns that were entirely defined by these elements, providing static decidability [3]. For instance, the Adapter pattern could be detected using this algorithm as it was defined entirely by the names of its participating classes and their relationships. However, the Singleton pattern cannot be detected through an algorithm that offers only static decidability, as the keywords ‘private’ and ‘static’ are essential to define this pattern.

Our presented tool addresses this limitation by utilizing an explicit specification that considers multiple features beyond class names and class relationships.

Our explicit specification for design patterns provides a complete definition for all 23 GoF design patterns, with the exception of the Strategy and State patterns, as they cannot be distinguished based solely on their structure and must be identified based on their intended purpose or meaning [4].

Based on our explicit specification for design patterns, our approach does exact subgraph matching. This method provides high precision in detecting patterns. Precision is defined as the number of true positives divided by the sum of true positives and false positives. In an educational setting, it is crucial to minimize the number of false positives, ideally to zero. This is because false positives can erode confidence in the detection software.

Using subgraph matching based on the explicit specifications, we can significantly reduce the occurrence of false positives. However, false positives can still occur if the templates match the Java sources but the functionality of the methods does not comply with the intent of the design pattern.

B. Explicit specification of design patterns

The starting point of our specification is, of course, the book of Gamma et al. [5], which describes design patterns using natural language, class diagrams, and examples of C++ code.

We summarize our explicit specifications. Details are given in [4]. Important specification elements are classes, attributes, operations, relationships, and modifiers.

The keyword ‘interface’ does not exist in C++ and OMT [12], which is used by Gamma et al. as a modeling technique but does exist in Java and UML. In many cases, an abstract class is equivalent to an interface. For the exceptions, ‘interface’ is added to the specification language of design patterns.

Dependency is a specification element that is used in Gamma et al. to denote class creating an object of another class [4].

In source code 1-N associations, aggregates and a composites are identically implemented, so they are described as 1-N associations [4].

An example of the explicit specification is Listing 1.

Listing 1. Template description of the Singleton pattern

```
<template name="Singleton">
  <class name="Singleton">
    <attribute name="uniqInstance" type="Singleton" modifier="private" isStatic="true"/>
    <operation name="getInstance" isStatic="true"/>
  </class>
</template>
```

C. Algorithm

The algorithm for detecting design patterns takes Java source code files as input, along with a file named ‘templates.xml’ that contains explicit specifications for each design pattern.

The Java sources are parsed by a parser generated by cup2 because of the produced parser’s speed [11], and the availability of a grammar of Java3. The Java parser generates information about the classes, interfaces, and their relationships. This results in a file named ‘inputSystem.xml’ with one template that has the same format as the templates in ‘templates.xml’. These two XML-files can easily be parsed by a SAX4 parser, which is part of Java.

For each design pattern in templates.xml, we search for a corresponding design pattern inputSystem.xml by a recursive “depth-first search” using two phases. Phase 1: In each call of the recursive “depth-first search”, an edge in the template we

1http://www.ptidej.net/tools/designpatterns

2http://www2.cs.tum.edu/projects/cup/index.php

3https://github.com/joewalnes/idea-community/tree/master/tools/lexer/jflex-1.4/examples/java

4http://www.saxproject.org/
are searching for, is matched with an edge in inputSystem.xml. The search ends when all edges are matched. Phase 2: Each class in the design pattern we are searching for, is compared with the corresponding class in inputSystem.xml by trying to match the attributes and methods of both classes.

To detect relations in Java sources, we adopt the definition of an association given by Guéhéneuc [6], neglecting run-time properties in his definition, because we use static detection. The definition becomes: An association between class A and B exists when an instance of class A can send a message to an instance of class B. Instances of classes occur as a field, array field, collection, parameter, and local variable.

IV. ASSESSMENT OF THE APPROACH

To assess the accuracy of our approach, we apply it to both a test set that we have constructed for the purpose of evaluating, and source code that is available “in the wild”. The research questions we will answer are: When given the design pattern specifications from section III-B:

RQ1 What is the accuracy of our detection software?
RQ2 What is the efficiency of our detection software?

The accuracy of the detection software can be determined by the ratio of correctly identified and rejected design patterns to the total number of searched design patterns. In cases where a design pattern is not detected, we will investigate the reasons for its failure to be detected. Furthermore, we will also address the occurrence of false positives in the detection results.

The efficiency is expressed as the total process time. During processing, Java files are read, parsed, and searched for design patterns.

The assessments are conducted on a PC with hardware characteristics Core™ i5-6400 CPU @ 2.70GHz x 4.

A. The test set

We have constructed a new test set comprising seven NetBeans projects, each containing a total of 23 distributed design patterns that align with the explicit specifications from section III-B. The test set has been designed to include classes that are associated with multiple design patterns, which increases the complexity of the search process. Furthermore, we ensured that some of the design patterns are not exact replicas of the GoF patterns. For instance, the Abstract Factory design pattern comprises three ConcreteFactories instead of the traditional two.

The results for running our tool on the test set are in Table I. The data shows that the design patterns can be processed within seconds. So the software is fast enough for an educational environment and therefore efficient.

Moreover, our proposed tool shows to be accurate because all design patterns, except the Facade pattern, can be detected. However, the detection software cannot distinguish between the State and Strategy pattern.

B. Java sources in the wild

We found two repositories for educational purposes with Java sources containing design patterns we will denominate RameshMF3 and sourcemaking6. These repositories offer for each design pattern a map with java sources. We made minor adjustments for the declaration of an attribute, keywords that we do not use, such as enum, module, and related keywords (as described in Section III-C)

The sources of RameshMF contain 18 of the 20 design patterns. This repository has no examples of the design patterns Interpreter, Mediator, Memento, and Visitor. For these design patterns, we used the sources of sourcemaking.

The results are shown in Table II.

<table>
<thead>
<tr>
<th>Design pattern</th>
<th>Lines of code</th>
<th>Detected</th>
<th>Total process time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RameshMF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Abstract Factory</td>
<td>567</td>
<td>Y</td>
<td>2.9</td>
</tr>
<tr>
<td>2 Adapter</td>
<td>330</td>
<td>Y</td>
<td>2.2</td>
</tr>
<tr>
<td>3 Bridge</td>
<td>759</td>
<td>N</td>
<td>4.1</td>
</tr>
<tr>
<td>4 Builder</td>
<td>430</td>
<td>N A</td>
<td>1.9</td>
</tr>
<tr>
<td>5 Chain of Responsibility</td>
<td>437</td>
<td>N A</td>
<td>1.9</td>
</tr>
<tr>
<td>6 Command</td>
<td>596</td>
<td>N</td>
<td>3.4</td>
</tr>
<tr>
<td>7 Composite</td>
<td>974</td>
<td>N</td>
<td>4.1</td>
</tr>
<tr>
<td>8 Decorator</td>
<td>404</td>
<td>N</td>
<td>3.9</td>
</tr>
<tr>
<td>9 Factory-pattern</td>
<td>932</td>
<td>Y</td>
<td>12.7</td>
</tr>
<tr>
<td>10 Flyweight</td>
<td>472</td>
<td>N A</td>
<td>2.3</td>
</tr>
<tr>
<td>11 Iterator</td>
<td>337</td>
<td>N A</td>
<td>1.7</td>
</tr>
<tr>
<td>12 Observer</td>
<td>591</td>
<td>N A</td>
<td>1.7</td>
</tr>
<tr>
<td>13 Prototype</td>
<td>602</td>
<td>Y</td>
<td>3.3</td>
</tr>
<tr>
<td>14 Proxy</td>
<td>275</td>
<td>N</td>
<td>2.1</td>
</tr>
<tr>
<td>15 Singleton</td>
<td>313</td>
<td>Y A</td>
<td>1.3</td>
</tr>
<tr>
<td>16 State</td>
<td>118</td>
<td>N</td>
<td>2.4</td>
</tr>
<tr>
<td>17 Strategy</td>
<td>61</td>
<td>N</td>
<td>1.9</td>
</tr>
<tr>
<td>18 Template Method</td>
<td>369</td>
<td>Y</td>
<td>3.0</td>
</tr>
<tr>
<td>sourcemaking</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19 Interpreter</td>
<td>147</td>
<td>N</td>
<td>1.0</td>
</tr>
<tr>
<td>20 Mediator</td>
<td>103</td>
<td>N</td>
<td>1.2</td>
</tr>
<tr>
<td>21 Memento</td>
<td>57</td>
<td>Y A</td>
<td>0.5</td>
</tr>
<tr>
<td>22 Visitor</td>
<td>82</td>
<td>N</td>
<td>1.8</td>
</tr>
</tbody>
</table>

The meaning of the values in column Detected is:

- A (Adjusted): A small change was made in the detection code to make detection possible. Examples: declaring an attribute public, and removing an enum definition.

3https://github.com/RameshMF/gof-java-design-patterns
6https://sourcemaking.com/design_patterns/


- N(o): The implementation of the design pattern matches the intent but not the class diagram as given by Gamma et al. [5]. Therefore, the implementation does not match the explicit specification of design patterns.
- Y(es): Detected without any problem.

The values in column Lines of code are the numbers of lines of Java code in the map containing the design pattern.

The results show the accuracy of our software because seven (5 * Y + 2 * Y A) detected design patterns are implemented according to their descriptions, and the implementations of the other 15 design patterns differ from their descriptions. For an educational environment, these results are accurate because the implementation of a design pattern has to match fully and not partially. The results are efficient because the total process time is at most 12.7 seconds.

V. CONCLUSIONS

Our main goal is to show that it is possible to build a design pattern detection tool using an explicit implementation.

The limitation of our tool is that we can make no distinction between State and Strategy patterns because that is impossible using static detection. Also, we chose not to include parts of the grammar that are less relevant to students, such as the keyword module, lambda expressions, and enum definitions.

Another limitation is that the Facade pattern cannot be detected. The specification of the Facade pattern in terms of classes and relations is simply too vague, too broad, to be usable.

Teachers could adjust the specification of the patterns to their own needs. The specification of design patterns is easily adaptable because it is written in XML and documented. Imaginable is, to make it possible to choose which language constructs are included or excluded.

For verification, we used a test set with 22 design patterns. The positive results of the applied test set contribute to the confidence of the specification of the design patterns and the quality of the software.

However, detection of the couple State and Strategy patterns implies a false negative.

But, if the source code of a method does not contain any statement then the intent of the class and template is not realized.

So, the detection of design patterns without false positives is not guaranteed.

For validation, we used several sources from the Internet. Design patterns that are present could sometimes not be recognized. The implementation of these patterns matched the intent but not the specification of the patterns.

The detection software is useful for an educational environment because the student’s elaboration of assignments has precisely to comply with the assignment.

The answers to our research questions are as follows:

1) The accuracy of the detection software is that all GoF design patterns, except the Facade pattern, can be detected when they fully match the defining template of the design pattern. No distinction can be made between the State and Strategy pattern. The software is not appropriate for searching for implementations of design patterns that only comply with the intent of a pattern.

2) The efficiency of the detection software depends on the number of lines of code and the design pattern. Java sources with less than 1000 lines of code are processed for all GoF design patterns within 12 seconds.

3) The new descriptions of design patterns are useful in an educational environment because a teacher may specify how design patterns should be implemented, and may allow only implementations that adhere to this specification.

In the future, we will try to improve the ease with which teachers may adapt the specification. Also, we may work on the feedback that the tool can give. Of course, we will also find out whether our tool can be really helpful, for teachers and/or students.

VI. ACKNOWLEDGEMENT

I thank Prof. Dr. Tanja Vos for her advice, discussions, and support.

REFERENCES

Fine-Grained Source Code Vulnerability Detection via Graph Neural Networks

1st Jingjing Wang  
National Key Laboratory of  
Science and Technology on Information System Security  
Beijing, China  
wang_1094412598@163.com

2nd Minhuan Huang  
National Key Laboratory of  
Science and Technology on Information System Security  
Beijing, China  
darbean@126.com

3rd Yuanpin Nie  
National Key Laboratory of  
Science and Technology on Information System Security  
Beijing, China  
yuanpingnie@nudt.edu.cn

4th Xiaohui Kuang  
National Key Laboratory of  
Science and Technology on Information System Security  
Beijing, China  
xiaohui_kuang@163.com

5th Xiang Li  
National Key Laboratory of  
Science and Technology on Information System Security  
Beijing, China  
ideal_work@163.com

6th Wenjing Zhong  
Xidian University  
Xi’an, China  
zhongwj@stu.xidian.edu.cn

Abstract—Although the number of exploitable vulnerabilities in software continues to increase, the speed of bug fixes and software updates have not increased accordingly. It is therefore crucial to analyze the source code and identify vulnerabilities in the early phase of software development. However, vulnerability location in most of the current machine learning-based methods tends to concentrate at the function level. It undoubtedly imposes a burden on further manual code audits when faced with large-scale source code projects. In this paper, a fine-grained source code vulnerability detection model based on Graph Neural Networks (GNNs) is proposed with the aim of locating vulnerabilities at the function level and line level. Our empirical evaluation on different C/C++ datasets demonstrated that our proposed model outperforms the state-of-the-art methods and achieves significant improvements even when faced with more complex, real-project source code.

Index Terms—deep learning, program analysis, vulnerability detection

I. INTRODUCTION

According to the report released by the National Institute of Standards and Technology (NIST) [1], the number of vulnerabilities found in 2022 is contributing to a sharp rise. The record number of vulnerabilities found over five consecutive years, along with the fact that bug fixes and software updates have not kept pace mean that we are now facing higher security risks than ever before. Consequently, in order to improve system security and code audit efficiency, as well as to further standardize programmers’ coding behavior, it is crucial to identify potential vulnerabilities in the programs and fix these in a timely fashion through source code analysis in the early stage of software development.

†Corresponding author  
DOI reference number: 10.18293/SEKE2023-115

The accuracy of conventional machine learning (ML) algorithms (i.e. Support Vector Machine, Decision Tree, Random Forest, etc.) for static source code analysis heavily depends on domain experts to perform feature engineering. However, this process becomes onerous and impractical as software source code scales up and functions become more complicated [2], [3]. Deep learning (DL) technology can overcome the drawbacks of conventional ML and automatically extract features from objects, provided that heuristic guidance strategies have been established. Nevertheless, source code is a structured language and deep neural networks often treat it as natural language in the feature extraction stage [4]–[7]. This results in the loss of program logic and structure information, limiting the DL model’s potential for vulnerability feature learning. In recent years, GNNs [8] have offered new insights into the static vulnerability analysis of source code by using intermediate representations such as abstract syntax trees (AST), control flow graphs (CFG), and data flow graphs (DFG) [9]–[12].

On the other hand, many ML-based vulnerability detection...
models are trained on datasets that contain synthetic samples to some extent. Nonetheless, due to dataset labeling and model granularity limitations, vulnerability location is commonly concentrated at the function level. In contrast, some studies have focused on more precise locations at the slice [13, 14] or line level [15]. However, these refined locations require strict data labeling requirements, as well as a laborious data preprocessing process.

In order to address the aforementioned issue, we propose a new vulnerability detection model based on GNNs that can accurately locate vulnerabilities at both the function and line level. The contributions of our work can be summarized as follows:

- We propose a novel GNN-based approach, which learns source code information through the intermediate representation of multidimensional program features. It is developed to improve the performance of both function-level and line-level vulnerability location and achieve efficient code auditing without the need for heavy manual engineering.

- We propose a vulnerability dataset with function-level and line-level labels, which were collected from popular open-source C/C++ projects, to further evaluate the effectiveness of our method. Compared with the existing public vulnerability datasets, our dataset is relatively more complete and valuable for further research.

II. RELATED WORKS

As the earliest DL-based vulnerability detection systems, Vuldeepecker [14] and Sysevr [13] utilize Bi-directional Long Short-Term Memory (BiLSTM) to apply fine-grained program representation in order to locate vulnerabilities at the slice level. Follow up studies included µVuldeepecker [16] and VulDeeLocator [17]. In addition, many studies extract code semantics based on AST and adopt vulnerability detection models in combination with BiLSTM [5], [18]–[20], which attempt to achieve high classification precision at the function level. Furthermore, the CPG was first proposed by [21], providing a new insight into source code vulnerability feature extraction. Some studies have realized function-level source code vulnerability identification based on GNNs [9], [10], [12]. These studies prove that these methods can effectively capture the program structure and node information carried by the CPG and its variants [11], [12], [22]. This compensates for the loss of important code logic and structural information in other deep learning models due to their use of a serialized feature learning process.

III. METHODOLOGY

Objective The goal of our vulnerability detection model is to predict the label \( y_i \in Y = \{0, 1\}^m \) of the CPG \( G_v \) corresponding to a given source code function \( C_i \in C \) with a mapping function \( f : G \rightarrow Y \). Here, \( C \) represents the set of source code function, while \( m \) is the total number of function instances; moreover, a vulnerable function is labeled with 1, and otherwise 0. To this end, our model is designed to learn an entire CPG representation \( h_g \) through a set of node representations \( \{H_v \mid v \in V\} \) obtained by a feature encoder that is used to decide a label \( f(G) = \hat{y} \); here \( v \) refers to the node feature vector, and \( \hat{y} \) is the prediction result. The mapping function \( f \) is then learned with a cross-entropy loss by minimizing the negative log-likelihood below:

\[
\min \sum_{i=1}^{m} -y_i \log \hat{y}_i.
\] (1)

The architecture of our model, illustrated in Figure 1, comprises the following three modules: 1) Embedding module. The Code Property Graphs (CPGs) are adopted as the intermediate representation of the source code. A multidimensional program feature encoding scheme is then designed to convert the CPGs into vectors, which forms the input of the model. 2) Location module. A novel location module is designed to capture important nodes of CPGs according to their IS value and return the corresponding potential vulnerable lines of code. 3) Classification module. BiLSTM is introduced as a readout function to generate the global representation of CPGs for identifying vulnerable functions.

A. Embedding Module

1) Code Property Graph Generation: Compared with using a single property, CPGs have been shown to be able to model more common vulnerability types [24], enabling it to achieve efficient vulnerability mining. As for the implementation, Jorn [21] is adopted to generate a joint data structure composed of code properties for source code.

2) Graph Embedding of Multidimensional Program Features: The node information of CPG consists of two parts: attributes and code. To encode the source code from various perspectives, such as function calls, logical operations, variable types, semantics, and syntax, a node compound feature embedding method has been devised in this stage. Figure 2 delineates the process of node feature embedding.

   a) Node Attribute Embedding: The node attribute feature consists of vectors of five fields. According to the tag, all the nodes are divided into different categories, representing the different roles played by nodes in the CPG. For example, the \( V_{op} \) field contains the encoding for predefined program operations, such as assignment, judgment, comparison, and so on. Similarly, the \( V_{func} \) field reflects the call relationship between the program and specific functions. Moreover, \( V_{lite} \) describes the variables involved in the operation of the program, such as characters and numbers, while \( V_{type} \) corresponds to 16 fixed parameter types in the C/C++ language. All the vectors of \( V_{attribute} \) are encoded via one-hot before being concatenated.

   b) Node Code Embedding: The semantic information of each node in the CPG is encoded through vectorization of the corresponding code statements. As for implementation, after cleaning the comments and removing non-ASCII characters, the code is normalized to alleviate the burden of feature encoding caused by the presence of numerous user-defined functions and variables independent of vulnerabilities. Finally, the tokenized code sequences are mapped to feature vectors based on the pre-trained Word2Vec model to obtain the fixed-size \( V_{code} \) and concatenate it with \( V_{attribute} \).
B. Location Module

1) GCN Layers: To aggregate the neighborhood information, we use graph convolutional network (GCN), first proposed by [25].

For a CPG $G_i$ with $n$ nodes and $d_v$ dimensional features, the definition of GCN is as follows:

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

(2)

Here, $H^{(l)}$ is the node representation of the $l$-th layer and is initialized by the node features matrix $X \in \mathbb{R}^{n \times d_v}$, $\tilde{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix with self-connections, $\tilde{A} = A + I_N$, $\tilde{D} \in \mathbb{R}^{n \times n}$ is the degree matrix of $\tilde{A}$, $W \in \mathbb{R}^{d_{in} \times d_{out}}$ is the weight matrix with input feature dimension $d_{in}$ and output feature dimension $d_{out}$, and $\sigma(\cdot)$ refers to the ReLU function [26], which is used as the activation function.

2) Line-level Location: To ensure that more attention is paid to the important nodes with high influence on vulnerabilities, the node score $Z \in \mathbb{R}^{n \times 1}$ is obtained by two-layer GCN learning, as follows:

$$Z(H, A) = \tanh(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)})$$

(3)

In the real world, differences between vulnerable and benign code may be subtle, but it is related to many nodes reflected in CPG, as shown in Figure 5. To make the node features after message-passing more distinguishable and attempt to capture more detailed vulnerability feature patterns, a learnable parameter matrix $\theta_i \in \mathbb{R}^{n \times 1}$ is introduced. Finally, the Influence Score $: IS \in \mathbb{R}^{n \times 1}$ of CPG nodes can be expressed, as follows:

$$IS(H, A) = LN(Z + \theta_i)$$

(4)

where $LN$ is a layer normalization [29].

On this basis, the $[kn]^{\text{th}}$ CPG nodes with the highest $IS$ would be retained; here, $k$ is the keep ratio. Subsequently, according to the graph mapping files (generated by Joern), the node indexes are mapped to the relevant lines of code in the source code file to locate the vulnerability on the line level. The locating process can be described as follows:

$$\hat{idx} = \text{top}_\text{rank}([IS, |kn|]),$$

(5)

$$\hat{H} = H_{\hat{idx}},$$

(6)

$$Loc = \text{map}(\hat{idx})$$

(7)

where $\text{top}_\text{rank}(\cdot)$, $\text{map} : \hat{idx} \rightarrow \text{line num}$ returns the indices of the retained nodes, $\hat{H} \in \mathbb{R}^{kn \times 1}$ is the new feature matrix used as the input of the next layer, and $Loc$ represents the set of line numbers of code mapped by $\hat{idx}$.

C. Classification Module

It is worth noting that there is often a strong correlation between multiple lines of code that contribute to a specific vulnerability, which in turn correspond to the key nodes in the CPG. However, some commonly used approaches to graph pooling [31], [32] ignore the interaction between nodes, or cause the loss of node information [33], [34]. For this purpose, when CPG is summarized as $[kn]^{\text{th}}$ important nodes, BiLSTM is introduced as a readout function that further considers the dependencies and inter-node relationships among these nodes to learn a $d_v$-dimensional meaningful graph representation $r_i \in \mathbb{R}^{d_v}$, as follows:

$$r_i = \text{BiLSTM}(\hat{H}).$$

(8)

Finally, the function-level prediction $\hat{y}_i$ is achieved through the two fully connected layers with softmax outputs, as follows:

$$\hat{y}_i = \text{Softmax}(W_F^{(2)}(W_F^{(1)} r_i + b^{(1)}) + b^{(2)}),$$

(9)

where $W_F^{(1)}$ and $b^{(1)}$ are parameters of the layer.
Fig. 3. (a): Example of a vulnerable function with an Out-of-bounds Read Error (CWE-125) from FFmpeg (unpatched); (b): The fixed vulnerable function (patched); (c): The simplified CPG of the vulnerable function in (a). The red nodes in (c) corresponding to the red line of code labeled as \textit{vulLoc} in (a).

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>COMPARISON OF FUNCTION-LEVEL CLASSIFICATION OF KNOWN CWE TYPES ON HD AND RD DATASETS. P: PRECISION(%); F1: F1-score(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>HD</td>
</tr>
<tr>
<td></td>
<td>CWE-119</td>
</tr>
<tr>
<td>ML-based</td>
<td>XGBoost</td>
</tr>
<tr>
<td></td>
<td>CNN</td>
</tr>
<tr>
<td>Models</td>
<td>Vuldeepecker [14]</td>
</tr>
<tr>
<td></td>
<td>Devign [12]</td>
</tr>
<tr>
<td>Commercial</td>
<td>Cpcheck [15]</td>
</tr>
<tr>
<td></td>
<td>Vuldeepecker [14]</td>
</tr>
<tr>
<td>Tools</td>
<td>RATS [37]</td>
</tr>
<tr>
<td></td>
<td>Flint++ [38]</td>
</tr>
<tr>
<td>Ours</td>
<td>98.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>COMPARISON OF FUNCTION-LEVEL CLASSIFICATION OF REAL-WORLD PROJECTS WITH UNKNOWN CWE TYPES ON RD DATASETS. P: PRECISION(%); F1: F1-score(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>RD</td>
</tr>
<tr>
<td></td>
<td>FFmpeg</td>
</tr>
<tr>
<td>ML-based</td>
<td>XGBoost</td>
</tr>
<tr>
<td></td>
<td>CNN</td>
</tr>
<tr>
<td>Models</td>
<td>Vuldeepecker [14]</td>
</tr>
<tr>
<td></td>
<td>Devign [12]</td>
</tr>
<tr>
<td>Commercial</td>
<td>Cpcheck [15]</td>
</tr>
<tr>
<td></td>
<td>Vuldeepecker [14]</td>
</tr>
<tr>
<td>Tools</td>
<td>RATS [37]</td>
</tr>
<tr>
<td></td>
<td>Flint++ [38]</td>
</tr>
<tr>
<td>Ours</td>
<td>82.1</td>
</tr>
</tbody>
</table>

IV. EXPERIMENTS

In this section, we conduct extensive experiments on two datasets to evaluate the effectiveness of the proposed model in performing fine-grained vulnerability location and compare it with that of state-of-the-art vulnerability detection methods.

A. Datasets

The experiments are carried out on two datasets: the Hybrid Dataset (HD) and the Real-project Dataset (RD).

1) Hybrid Dataset (HD): In order to verify the impact of different levels of dataset complexity on the performance of vulnerability detection methods, along with the gap between these methods and practice, the dataset proposed by VulDeepecker [14] is utilized in the experiments, which includes two known vulnerabilities: Memory Buffer Errors (CWE-119) and Resource Management Errors (CWE-399).

2) Real-project Dataset (RD): The statistics presented in [10] indicate that samples from real projects constitute a minor proportion of the dataset HD. To ensure that our proposed model effectively detects vulnerabilities in real software applications and makes meaningful contributions to production-level code security audits, we collected a completely real project dataset, RD.

RD contains 17752 programs with function-level and partial line-level data labeling for 13 popular C/C++ libraries and is available at https://github.com/fgVDgnn/fgVDgnn. Functions and lines of code corresponding to the security commits from NVD are labeled according to the version changes before and after the patch. As shown in Figure 3, compared with patched function (3(b)), we label patch-related statements in vulnerability functions (3(a)) as a vulnerable line of code (\textit{vulLoc}).

B. Baselines

We select two categories of methods in the field of static source code vulnerability analysis for performance comparison, as follows. 1) ML-based vulnerability detection models: XGBoost, CNN, VulDeepecker [14] and Devign [12]. We conduct experiments on the reproducible version of these methods to evaluate the performance of our model compared...
with typical ML-based methods. 2) Commercial code analysis tools: Cppcheck [35], Flawfinder [36], RATS [37] and Flint++ [38]. They are popular commercial tools for scanning code and reporting potential security vulnerabilities, which can be used as a simple guide to static source code analysis.

C. Results

1) Results on function-level classification: For the function-level classification task, we conduct experiments on HD and RD respectively to explore the effect of the vulnerability detection methods when facing source code with known types of vulnerability, along with the impact of different levels of data complexity on their performances. Furthermore, experiments are carried out on different projects of unknown vulnerability types on our proposed RD dataset to evaluate the effectiveness of different detection methods in practical applications. The experimental results are reported in Tables I and II.

In summary, our proposed model achieves state-of-the-art performance and significant improvements on both datasets. In particular, when faced with the sophisticated real-project samples from RD, the efficiency of almost all methods can be seen to significantly decrease; however, the relative precision (P) and F1 score gains achieved by our model is an average of 17.0% and 12.0%. It is further demonstrated that our proposed model can still maintain good vulnerability detection performance compared with other methods in practical applications.

2) Line-level Location Results: The line-level localization performance of the model is evaluated on four projects contained in our proposed RD; more detailed statistics are shown in Table III.

As is evident, vulnerability lines account for only a very small part of a program, and our goal is to more efficiently implement source code security audits during the software development phase. Therefore, we introduce two indicators of HitRate and vulnerability line Coverage of the model from the perspective of graph, which are expressed as follows:

\[
\text{HitRate} = \frac{\sum_{i=0}^{m} n_{\text{hits}}}{\sum_{i=0}^{m} \lceil kn \rceil},
\]

\[
\text{Coverage} = \frac{\sum_{i=0}^{m} n_{\text{hits}}}{\sum_{i=0}^{m} n_{\text{vul}}},
\]

where \(n_{\text{hits}}\) represents the number of nodes correctly predicted by our model, and \(n_{\text{vul}}\) is the number of nodes related to the vulnerability code in each CPG. We graph the experimental results in Figure 4.

In addition to function-level vulnerability location, we further divide three location ranges according to AFL to verify the effectiveness of the method under different granularities. It can be observed that, in most cases, the model’s coverage of vulnerability lines can be maintained at a high level. Furthermore, a vulnerability location within 50 lines can reduce the amount of code required for function-level code auditing by at least 47.9% while still maintaining a relatively promising hit rate.

V. Conclusion

In this paper, we propose a novel GNN-based source code vulnerability detection model designed to achieve fine-grained potential vulnerable code identification at a function level and line level through the intermediate representation of multidimensional program features. Extensive experiments reveal the superior performance of our model compared with other state-of-the-art methods. It is further demonstrated that our approach can be applied to support the source code vulnerability detection of real projects, which greatly reduces the workload associated with manual code audits.

REFERENCES

Software Defect Prediction via Positional Hierarchical Attention Network

Xinyan Yi¹, Hao Xu¹, Lu Lu¹,³*, Quanyi Zou²,³, Zhanyu Yang¹
¹School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
²School of Journalism and Communication, South China University of Technology, Guangzhou, China
³Pazhou Laboratory, Guangzhou, China
*Corresponding author email: lul@scut.edu.cn

Abstract—Software Defect Prediction (SDP) aims to identify defect-prone modules in advance to ensure software quality. In SDP research based on deep learning, the mainstream approach is to extract deep semantic features from an Abstract Syntax Tree (AST). Theoretically, the AST as a bi-dimensional structure encloses information at the node level, fragment level, and entire tree level. However, most existing research serializes the whole AST without considering the expression at different granularities. To address this limitation, we introduce a positional hierarchical attention network (PHAN) that acquires semantic features by simultaneously considering contexts between nodes and paths. Specifically, our model incorporates attention mechanisms to capture information of varying importance at separate hierarchies, and relative position representations to distinguish the contributions of different paths. Experimental results demonstrate that PHAN significantly outperforms existing baseline methods.

Keywords—software defect prediction; abstract syntax tree; hierarchical attention; deep semantic feature

I. INTRODUCTION

As the scale and complexity of software continue to boost, software defects pose greater challenges to software quality. SDP is the technique that leverages historical data to identify defect-prone software modules in advance, enabling the enhancement of software quality and testing efficiency.

Early research in SDP using traditional hand-crafted features is limited to capturing semantic information. With the emergence of deep learning, some studies applied natural language processing methods to encode codes as general texts [1]. However, symbols like arithmetic operators may involve useless ambiguous information. AST as a representation of the code’s syntax structure, inherently encapsulating the syntax structure and semantic information, has shown its promising performance in numerous code-related tasks [2]. In SDP tasks, AST is commonly presented through traversal [3], tree-based networks [4], and graph-based networks [5].

However, previous research methods on semantic encoding in SDP still have limitations. Firstly, whether AST is flattened as a text sequence or maintains its original structure, they encode the AST as a whole, ignoring the information at a moderate granularity level. Secondly, although some studies in SDP have taken a hierarchical structure into account, such as decomposing code into tokens and lines levels [1], and splitting AST into nodes and subtrees levels [6], there is a lack of encoding at the path granularity in SDP. Furthermore, existing AST presentation by mining paths [7] does not consider the positional information between paths, while the positional difference may indicate the existence of defects.

In this paper, we propose a positional hierarchical attention network for SDP. The main contributions are as follows:

1) Design a method for learning AST representation based on hierarchical path mining and tree reconstruction.
2) Introduce relative position encoding at the path level to alleviate the lack of long-distance path dependencies in SDP.

II. RELATED WORKS

A. Code Representations in Defect Prediction

Majd et al. represented code in the forms of tokens, lines, and statements, which focus on the code’s high level [8]. Wang et al. transformed the source code into AST and extracted vectors through pre-order traversal of the AST [9]. Phan et al. argued that the Control Flow Graph can better represent code information in SDP [10]. Tian et al. extracted code slices and converted them into System Dependency Graphs for control flow and data flow analysis [11]. Chen et al. transformed each character into a pixel, visualizing the programs as images [12].

B. Deep Learning Approaches for Defect Prediction

Wang et al. suggested using Deep Belief Networks (DBNs) and source code changes to capture semantic feature representations [9]. Then, Li et al. combined semantic features extracted by Convolutional Neural Networks (CNN) with hand-crafted features, which improved prediction performance by exploiting their superior ability to mine local features [13]. Dam et al. leveraged a tree-structured Long Short-term Memory (LSTM) network to automatically mine long context dependency in code [4]. Uddin et al. employed Bi-LSTM to learn contextual information from the token vectors embedded through the pre-trained model BERT [14].

III. METHOD

In this section, the framework of the proposed model is illustrated in Fig. 1. During the preprocessing stage, JAVA files are transformed into ASTs, simultaneously tracking the paths of trees. In the encoding stage, semantic features are extracted hierarchically at both the node and path levels. Lastly, the resulting AST encoding is subsequently fed into a classifier.

DOI reference number: 10.18293/SEKE23-119
A. Source Code File Preprocessing
Prior to encoding, toolkit javalang\(^1\) parses the Java code into an AST. Then, all root-to-leaf paths are extracted by a traversal, and nodes are initialized as vectors through an embedding layer:

\[
nv_{it} = \text{Embedding}(n_{it}), t \in [1, T], i \in [1, L],
\]  

where \(nv_{it}\) denotes an initial vector of the \(t\)-th node in the \(i\)-th path, \(T\) is the number of nodes in the longest path, \(L\) is the number of paths in the biggest AST.

B. Node Level Encoder
At the node level, the additive attention [15] is adopted. **Node additive attention.** Given the set of preliminary vector representation \(N = [nv_{i1}, nv_{i2}, ..., nv_{iT}]\), the annotations of nodes are firstly obtained through a bidirectional GRU (Bi-GRU) [16] layer. We get the node annotation \(h_{i,t}\) incorporating node contextual information by concatenating the output of forward direction \(h_{i,t}^f\) and the backward direction \(h_{i,t}^b\):

\[
\begin{align*}
\overrightarrow{h_{i,t}} &= \text{GRU}(nv_{it}), t \in [1, T], \\
\overleftarrow{h_{i,t}} &= \text{GRU}(nv_{it}), t \in [T, 1], \\
h_{i,t} &= [\overrightarrow{h_{i,t}}, \overleftarrow{h_{i,t}}].
\end{align*}
\]

Bi-GRU has the potential to learn positional information, making additional position information unnecessary. Subsequently, the node annotation \(h_{i,t}\) is directly fed into a single-layer perceptron to obtain its nonlinear representation \(u_{i,t}\):

\[
u_{i,t} = \tanh(W_n h_{i,t} + b_n).
\]

Then the similarity of a node-level contextual vector \(u_n\) with each hidden representation \(u_{i,t}\) is computed as importance metrics and normalized through a softmax function to obtain an importance weight \(\alpha_{i,t}^N\):

\[
\alpha_{i,t}^N = \text{softmax} \left(u_{i,t}^T u_n\right),
\]

where \(u_n\) is initialized as a trainable parameter.

\(^1\)https://github.com/c2nes/javalang

Path encoding. All the critical node information is aggregated by taking a weighted sum of the hidden states and attention weights, resulting in the entire path encoding \(pv_i\):

\[
pv_i = \sum_t^{T} \alpha_{i,t}^N h_{i,t}. \tag{7}
\]

C. Path Level Encoder
Hierarchical Attention Network (HAN) [17] leverages a shared RNN encoder and attention mechanism for word-level and sentence-level. However, in SDP, the number of paths \(L\) is significantly larger than the number of nodes \(T\), resulting in a long sequence where Bi-GRU is limited to capturing long-range dependencies due to the gradual decay. Thus we propose a positional self-attention mechanism at the path level. **Path positional self-attention.** Given the set of path vector \(P = [pv_1, ..., pv_L]\), self-attention[4] can be defined as:

\[
\text{SelfAtt}(Q, K, V) = V \text{softmax} \left(\frac{K^T Q}{\sqrt{d}}\right)
\]

\[
K = W_K P, Q = W_Q P, V = W_V P,
\]

where \(d\) denotes the dimension of each path vector \(pv\), \(W_K\), \(W_Q\) and \(W_V\) are parameter matrices.

Self-attention does not consider the positions which often carry semantic meaning, so we introduce relative position embedding [18]. The self-attention score between the \(i\)-th and \(j\)-th path is measured by the scaled dot product of \(q_i \in Q\) and \(k_j \in K\). In absolute positional encoding, the positional embedding \(pos\) is added straightforwardly to the input vector, then the self-attention weight \(\alpha_{i,j}^P\) is calculated as:

\[
\alpha_{i,j}^P = \text{softmax} \left(\frac{k_j^T q_i}{\sqrt{d}}\right)
\]

\[
= \text{softmax} \left(\frac{(W_K p_{v_j} + W_K p_{pos_j})^T (W_Q p_{v_i} + W_Q p_{pos_i})}{\sqrt{d}}\right). \tag{9}
\]

In relative positional encoding, the position of each path is not explicitly represented, instead, the relative distance between the current position and the position being attended to is taken into account. We eliminate the \(W_Q p_{pos}\) and replace

\[
\alpha_{i,j}^P = \text{softmax} \left(\frac{k_j^T (p_{v_j} - p_{v_i})}{\sqrt{d}}\right)
\]

\[
= \text{softmax} \left(\frac{(W_K p_{v_j} + W_K p_{pos_j})^T (W_Q p_{v_i} + W_Q p_{pos_i} - W_Q p_{pos_j})}{\sqrt{d}}\right).
\]

Fig. 1. The framework of PHAN.
the $W_{K\text{pos}_j}$ with a binary positional vector $R_{i,j}^\alpha$ and weighted by multiplication with the attention score $\alpha_{i,j}$ to obtain the output $o_i$:

$$o_i = \sum_j \alpha_{i,j}^P v_j = \sum_j \alpha_{i,j}^P (W_P pv_j + W_{V\text{pos}_j}) = \sum_j \alpha_{i,j}^P (W_P pv_j + R_{i,j}^\alpha).$$

**AST encoding.** All paths are aggregated by taking a weighted sum of the outputs $O = [o_1, ..., o_L]$, resulting in the AST encoding $astv$:

$$astv = OW_{ast} = \sum_{i=1}^{L} \beta_i o_i, \quad W_{ast} = [\beta_1, ..., \beta_L]^T,$$

where $W_{ast} \in \mathbb{R}^{L \times 1}$ denotes a trainable parameter matrix.

**D. Software Defect Prediction via PHAN**

This AST vector $astv$ is utilized as the semantic feature and fed into a Logistic Regression (LR) for defect prediction.

**IV. EXPERIMENTAL SETTING**

**A. Experimental Datasets**

The dataset employed is the classic SDP repository PROMISE [19], detailed information is presented in Table I. This paper chooses the old version source code used for training and the new one for testing.

**B. Evaluation Metrics**

Higher F-measure values indicate more robustness. MCC provides a better evaluation when dealing with imbalanced distributions. AUC represented as an area under the curve can avoid experimental errors. Specifically,

$$\text{Precision} = \frac{tp}{tp + fp}, \quad \text{Recall} = \frac{tp}{tp + fn}$$

$$\text{F - measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{MCC} = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}},$$

where $tp, fp, tn$, and $fn$ refer to True Positive, False Positive, True Negative, and False Negative, respectively.

**C. Comparison Methods**

PHAN is compared with the following methods:

- **LR:** A traditional classification method utilized to predict defect-proneness based on hand-crafted features.
- **DBN-WP [9]:** A standard DBN model extracting semantic features from AST for within-project defect prediction.
- **DP-CNN [13]:** An enhanced CNN method based on hand-crafted features and deep semantic features for SDP.
- **DP-LSTM [3]:** A Bi-LSTM Model extracting semantic representations for SDP.
- **HAN [17]:** Original hierarchical attention network at node and path level for SDP.

**V. EXPERIMENTAL RESULTS**

This section aims to show comparisons of the proposed method with baselines and analyze the experimental results.

In this study, we investigate the effectiveness of the PHAN compared to a traditional model LR based on hand-crafted features and baseline models based on deep learning covering DBN, CNN, and LSTM. Table II presents that the proposed method outperforms other baseline models in 6 out of 9 F-measure values, 8 out of 9 AUC values, and 7 out of 9 MCC values. Experimental results indicate that PHAN has achieved superior performance compared with a traditional method and several advanced AST-based methods in multiple perspectives. Furthermore, we compare the performance of the proposed PHAN and HAN on SDP tasks. Table II shows that PHAN exceeds HAN across 7 projects. While HAN achieves comparable performance to PHAN on most projects, PHAN performs particularly well on the Xerces project, compensating for the poor performance of HAN on this project.

The line chart in Fig. 2 presents a visual comparison of the proposed method and the baseline models, illustrating that PHAN outperforms other models in most cases, particularly in the Ant and Xerces projects. As revealed by the average defect rate of each project shown in Table I, the Ant and Xerces projects of the old version have severe data imbalance problems, posing challenges for learning valuable information from the training set. Notably, all deep learning models exhibit significantly worse performance on the two projects, while only the traditional machine learning model LR achieves similar performance to PHAN. These results highlight that representing ASTs at a finer-grained level and distinguishing the contribution of different paths’ positions can learn more effectively from imbalanced data distributions.

**VI. CONCLUSION**

This study presents PHAN, a positional hierarchical attention network model designed to extract code semantic features from ASTs. The PHAN encodes at a finer-grained path level, taking into account the contributions of different nodes and paths to SDP tasks. This approach is empirically compared with traditional methods, state-of-the-art deep learning methods, and original HAN across nine Java projects. Our findings demonstrate that the proposed model achieves the best performance across F-measure, AUC, and MCC values.
TABLE II. F-MEASURE, AUC, MCC COMPARISON OF DIFFERENT MODELS

<table>
<thead>
<tr>
<th>Method</th>
<th>Metric</th>
<th>Ant</th>
<th>Jedit</th>
<th>Log4j</th>
<th>Lucene</th>
<th>Poi</th>
<th>Synapse</th>
<th>Velocity</th>
<th>Xalan</th>
<th>Xerces</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F-measure</td>
<td>0.553</td>
<td>0.511</td>
<td><strong>0.667</strong></td>
<td>0.619</td>
<td><strong>0.729</strong></td>
<td>0.512</td>
<td>0.544</td>
<td>0.524</td>
<td>0.519</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td>0.702</td>
<td>0.693</td>
<td><strong>0.746</strong></td>
<td>0.584</td>
<td>0.681</td>
<td>0.612</td>
<td>0.636</td>
<td>0.565</td>
<td>0.628</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td>0.361</td>
<td>0.329</td>
<td><strong>0.474</strong></td>
<td>0.163</td>
<td>0.348</td>
<td>0.212</td>
<td>0.260</td>
<td>0.130</td>
<td>0.272</td>
</tr>
<tr>
<td>DBN-WP</td>
<td>F-measure</td>
<td>0.416</td>
<td>0.537</td>
<td>0.570</td>
<td>0.413</td>
<td>0.570</td>
<td>0.465</td>
<td>0.541</td>
<td>0.482</td>
<td>0.538</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td>0.625</td>
<td>0.663</td>
<td>0.682</td>
<td>0.570</td>
<td>0.615</td>
<td>0.582</td>
<td>0.610</td>
<td>0.560</td>
<td>0.577</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td>0.225</td>
<td>0.163</td>
<td>0.224</td>
<td>0.222</td>
<td>0.182</td>
<td>0.186</td>
<td>0.172</td>
<td>0.185</td>
<td>0.191</td>
</tr>
<tr>
<td>DP-CNN</td>
<td>F-measure</td>
<td>0.404</td>
<td>0.546</td>
<td>0.538</td>
<td>0.541</td>
<td>0.438</td>
<td>0.520</td>
<td>0.560</td>
<td>0.543</td>
<td>0.547</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td>0.600</td>
<td>0.632</td>
<td>0.697</td>
<td>0.605</td>
<td>0.609</td>
<td>0.602</td>
<td>0.645</td>
<td>0.607</td>
<td>0.498</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td>0.207</td>
<td>0.169</td>
<td>0.147</td>
<td>0.173</td>
<td>0.152</td>
<td>0.158</td>
<td>0.240</td>
<td>0.119</td>
<td>0.168</td>
</tr>
<tr>
<td>DP-LSTM</td>
<td>F-measure</td>
<td>0.443</td>
<td>0.518</td>
<td>0.566</td>
<td>0.619</td>
<td>0.719</td>
<td>0.246</td>
<td>0.505</td>
<td><strong>0.610</strong></td>
<td>0.550</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td>0.527</td>
<td>0.681</td>
<td>0.644</td>
<td>0.602</td>
<td>0.516</td>
<td>0.603</td>
<td>0.619</td>
<td>0.639</td>
<td>0.571</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td>0.227</td>
<td>0.371</td>
<td>0.328</td>
<td><strong>0.268</strong></td>
<td>0.235</td>
<td>0.213</td>
<td>0.352</td>
<td>0.259</td>
<td>0.234</td>
</tr>
<tr>
<td>HAN</td>
<td>F-measure</td>
<td>0.527</td>
<td>0.553</td>
<td>0.632</td>
<td>0.652</td>
<td>0.713</td>
<td>0.547</td>
<td>0.596</td>
<td>0.595</td>
<td>0.474</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td>0.679</td>
<td>0.724</td>
<td>0.716</td>
<td><strong>0.628</strong></td>
<td>0.684</td>
<td>0.648</td>
<td>0.680</td>
<td>0.641</td>
<td>0.604</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td>0.325</td>
<td>0.395</td>
<td>0.416</td>
<td>0.250</td>
<td>0.354</td>
<td><strong>0.285</strong></td>
<td>0.342</td>
<td>0.286</td>
<td>0.228</td>
</tr>
<tr>
<td>PHAN</td>
<td>F-measure</td>
<td><strong>0.568</strong></td>
<td>0.561</td>
<td>0.645</td>
<td>0.632</td>
<td>0.703</td>
<td>0.545</td>
<td><strong>0.603</strong></td>
<td>0.589</td>
<td>0.674</td>
</tr>
<tr>
<td></td>
<td>AUC</td>
<td><strong>0.715</strong></td>
<td>0.736</td>
<td>0.727</td>
<td>0.627</td>
<td><strong>0.698</strong></td>
<td>0.639</td>
<td>0.690</td>
<td>0.644</td>
<td>0.667</td>
</tr>
<tr>
<td></td>
<td>MCC</td>
<td><strong>0.385</strong></td>
<td>0.405</td>
<td>0.436</td>
<td>0.249</td>
<td><strong>0.381</strong></td>
<td>0.264</td>
<td><strong>0.363</strong></td>
<td>0.295</td>
<td><strong>0.322</strong></td>
</tr>
</tbody>
</table>

(a) AUC values comparison.  
(b) F-measure values comparison.  
(c) MCC values comparison.

Fig. 2. The results of diverse metrics among baseline models and PHAN.

VII. ACKNOWLEDGMENT

This work was supported by the second batch of cultivation projects of Pazhou Laboratory in 2022, No. PZL2022KF0008.

REFERENCES


Abstract—To efficiently fix defects within a specific time frame during software development, researchers have proposed defect severity prediction to help developers determine which defects to fix first and make efficient use of limited resources. Additionally, to improve the efficiency of defect fixing, just-in-time (JIT) defect prediction has been proposed to promptly predict code fragments that may introduce defects when developers make code changes (i.e., submit a commit). In this way, defect feedback is prompt and localization precision is high. Typically, high-priority defects must be addressed as soon as possible, but when the bug report records a defect back to developers, they need to take time to get reacquainted with the related code fragments, slowing down the speed of high-priority defect fixes. Therefore, we used three machine learning algorithms to develop a JIT defect severity prediction model that allows developers to classify the severity of potential defects when submitting code changes. Our models were tested on ten large-scale projects and showed they can effectively predict defect severity just in time. With Random Forest, our models achieved an average precision of 0.552, and an average recall, F1-measure, and AUC of 0.579, 0.528, and 0.729, respectively. Using Decision Tree, the average precision, recall, F1-measure, and AUC achieve 0.479, 0.494, 0.485, and 0.619, respectively; Using KNN, the average precision, recall, F1-measure, and AUC are 0.466, 0.468, 0.467, and 0.593, respectively. Meanwhile, we find a large portion (90.3% on average) of defect-introducing changes are at a high severity level.

Index Terms—JIT Defect Severity Prediction; Defect-fixing Changes; Defect-introducing Changes

I. INTRODUCTION

During software development and maintenance, repairing software defects is an indispensable task, and it often needs to be completed within a specified time period [1]. Hence developers usually assign a severity level to different defects, so that developers can give priority to fixing the high-severity defects when given limited time [1], [2]. In addition, the defect severity level also helps to reasonably assign defect-repairing tasks to the developers with appropriate development experience, which could avoid incorrect repairs [3]. However, classifying defect severity is mainly done manually by testers relying on their own expertise and experience [2], [4]. For software systems with massive defects, the manual assessment of defect severity would be time-consuming and error-prone [4], [5]. Therefore, more and more researchers have paid attention to defect severity prediction and explored approaches for automatically predicting the severity of defects [1]–[7].

JIT defect prediction has become an active research area [8]–[11], which predicts code fragments that may introduce defects when developers make a code change (i.e., submit a commit) [8]. Compared with the defect prediction based on the coarse-grained level of modules, packages, files, etc., this change-level prediction can narrow the scope of the location of the defects introduced [8], so that developers can quickly review potential defects, and each change has a unique author [8], so it is easy to determine the developers who introduced the defects. In addition, the defects are predicted before the changes are submitted, so developers can immediately repair the code when they are very familiar with the code fragments [8], saving time in recalling the implementation decisions.

In this paper, we introduced the concept of JIT defect prediction into the prediction of defect severity. Unlike traditional methods, a defect often got documented and classified several weeks or months later than it was introduced. Consequently, developers may need to take time to get reacquainted with the related code fragments for analyzing or even fixing the defect. With the JIT prediction on defect severity, developers would be able to promptly classify the severity of potential defects when committing code changes. This would help developers quickly assess the severity of defects and allocate resources for further repairs. Especially for the high-severity defects, which need to be fixed as soon as possible, developers can quickly review code snippets and fix them, since they are familiar with the code, which saves time in recalling the code and speeds up the repair of high-priority defects.

In this study, we use Random Forest, Decision Tree, and KNN classifier to establish JIT defect severity prediction model and apply the model to ten large-scale projects. The results show that our models can effectively predict defect severity just in time. The Random Forest-based model can get a precision of 0.552 on average, and the average recall, F1-measure, and AUC are 0.579, 0.528, and 0.729, respectively. For the Decision Tree-based model, the average precision, recall, F1-measure, and AUC are 0.479, 0.494, 0.485, and 0.619, respectively. When using KNN, the average precision, recall, F1-measure, and AUC can reach 0.466, 0.468, 0.467, and 0.593, respectively. In addition, we find a large portion (90.3% on average) of defect-introducing changes are at a high severity level.

The rest of the paper is structured as follows: Section II introduces the background of this paper. Section III reports the related work. Section IV introduces the methodology of our model building. Section V presents the experimental findings. Section VI describes the shortcomings of this research and future work. Section VII concludes.
II. BACKGROUND

A. Bug Report

During software development and maintenance, issue tracking systems are often used to record and track possible change requests, such as feature addition, bug fix, improvement, etc., in which the bug-fixing information is recorded in the bug report. The format of bug reports may vary from the issue tracking system, but there is always some common information in a bug report, including the ID, description, severity of a bug, and the products or components affected by the bug [2].

B. Defect Severity

The defect severity level is used to characterize the degree of negative impact by software defects [1]. Commonly used issue tracking systems always have a field to denote the severity of a defect in the report. This paper describes the severity level through the Priority field provided by Jira. We consider Blocker, Critical, and Major as high severity levels similar to [4], while Minor and Trivial are considered low severity levels.

C. Defect-fix change and Defect-introducing change

A defect-fix change is to fix a defect as recorded in the issue tracking system. Following the prior studies [12], we matched the committed changes with bug reports to identify defect-fix changes: if a commit’s message references a bug id, we considered it to be a defect-fix change. A defect-introducing change is a change that introduces a defect that needs to be fixed in the future during a code change [13], [14]. In this paper, we identify defect-introducing changes by using the SZZ algorithm Section II-D elaborated in the next section.

D. SZZ Algorithm

Śliwerski et al. [13] first proposed the SZZ algorithm to automatically identify changes as defect-introducing or non-defect-introducing, which has played a critical role in JIT defect prediction. This algorithm mainly consists of these steps: 1) matching the bug reports with the commits recorded in revision history to identify defect-fix commits; 2) using the diff command of the version control system to compare a defect-fix commit with its previous commit to get the changed code fragments, i.e., defect-fix code; 3) using the annotate command to retrace the history of defect-fix code fragments to locate the nearest commit where the code was added. Then this commit is considered as the commit inducing the defect-fix code, thus it is a candidate for the defect-introducing commit (change); 4) filtering out the candidates whose defect-introducing time is later than the defect-fix time. Since a defect should be introduced first, then it can be fixed later [13].

III. RELATED WORK

Our work is mainly related to JIT defect prediction and defect severity prediction. We will introduce some prior studies from these two aspects.

1 https://www.atlassian.com/software/jira

JIT defect prediction. Kamei et al. [8] applied SZZ algorithm [13] to mark code changes as defect-introducing or non-defect-introducing, extracted 14 metrics from the dimensions of Diffusion, Size, Purpose, History, and Experience and built a logistic regression model for JIT defect prediction, which showed 68% accuracy and 64% recall. Yang et al. [10] extracted the same features as Kamei et al. [8] from changes with known labels and employed a two-layer ensemble learning approach TLEL for JIT defect prediction. The result showed that TLEL was more effective than employing a single machine learning algorithm and it requires only checking 20% of the lines of code to discover over 70% of the defects. Borg et al. [15] utilized the SZZ algorithm [13] to label code changes, extracted metrics from various dimensions, such as Diffusion, Size, Experience, etc, and built a random forest model for JIT defect prediction. The researchers also made the SZZ algorithm open-source. In addition to the above supervised learning algorithms, researchers [16]–[18] have leveraged unsupervised learning algorithms to predict defects just in time without labeling.

Defect severity prediction. Lamkanfi et al. [6] compared Naive Bayes, Naive Bayes Multinomial, Support Vector Machines, and K-Nearest Neighbour in severity prediction, and presented that Naive Bayes Multinomial is the most effective for classifying severe and non-severe bug reports in two open-source systems, Eclipse and GNOME. Sahin and Tosun [7] found that using word embeddings for feature extraction and constructing models using Convolutional Neural Networks (CNN), Long Short Term Memory (LSTM), and Extreme Gradient Boosting (XGBoost) algorithms can effectively predict defect severity. Meanwhile, they found that word embeddings and deep learning techniques can also be used to directly predict the severity score of defects. Arokiam et al. [5] utilized previous bug report writing styles as features to predict defect severity, which outperformed existing keyword-based methods and can detect defects early in new projects.

In recent years, researchers have proposed various studies for JIT defect prediction or defect severity prediction, but none of them has focused on JIT defect severity prediction. Our paper is the first attempt to address this research gap. When developers submit changes, it is helpful to predict which change may introduce defects and how severe the defects are. Thus we conduct JIT defect severity prediction models in this work.

IV. METHODOLOGY

This section focuses on our experimental approach. The flow chart of our method is shown in Fig. 1. It mainly includes the steps of labeling, feature extraction, and model building.

A. Labeling

In this paper, we used the tool of SZZ Unleashed 2 by Borg et al. [15] to obtain the commit Ids of all defect-fix and defect-introducing change pairs. Then we traversed all

2 https://github.com/wogscpar/SZZUnleashed
commits in a project’s revision history to identify the defect-introducing commits: if a commit’s id is not in the identified Ids, we label this commit as 0, which means that this commit does not introduce defects. For a defect-introducing commit, we first get its paired defect-fix commit. Then based on the bug Id in the defect-fix commit’s message to analyzed the report of this defect to determine its severity. Following the prior studies, Minor and Trivial are considered to be low severity levels [4], since there is little difference in severity between them, thus we labeled both of them to be 1, Major is labeled as 2, Critical is labeled as 3, and Blocker is labeled as 4. The greater the value, the more severe the defect. If a commit introduced multiple defects, we labeled it based on the defect with the greatest severity.

B. Feature Extraction

Table I lists the features we extracted to represent each change. The first column shows the types of selected features; the second column is each feature; the third column shows the description of each feature. These fourteen features of the five dimensions used in this paper are widely used in JIT defect prediction [8]–[11].

Size dimension measures the size of the change. Larger code modifications are more likely to introduce defects, and relative code churn metrics are better defect predictors than absolute metrics [19], so we standardize the LA, LD, and LT by dividing LA and LD by LT and dividing LT by NF, as in [15].

Diffusion is used to measure the distribution of changed code in related files. When code change is spread across multiple related files, it can increase complexity and introduce potential defects [20]. We calculate these four features using the method described in [8].

Purpose takes the purpose of the modified code fragments into consideration. Defects are more likely to be introduced in the process of repairing [13], so many studies [9]–[11], [21] have used this type of feature for JIT defect prediction. For the FIX, we define its value based on whether the current change fixes the defect and the severity of the defect being fixed. If the change is not committed to fix the defect, FIX is 0. Otherwise, when the severity of the defect repaired by the change is Minor or Trivial, Major, Critical and Blocker, the FIX is 1, 2, 3, and 4, respectively.

History dimension measures the modification history of files that the current change is about to modify. The more times related files are modified or modified by more developers or the smaller the time difference between the latest change and this change in a relevant file, the more likely there are defects [8], [22]. Therefore, this dimension is commonly used for JIT defect prediction [9], [11], [15].

Experience dimension measures the experience of developers making changes. The more experienced the developer who changes the code, the less likely it is to introduce defects [21], so we use the features from this dimension for JIT defect prediction, similarly to [10], [11], [23].

C. Model Development

Machine Learning Algorithms. We selected three machine learning algorithms for building our prediction models. All of them have been widely used in defect predictions and support multi-classifications, with their effectiveness having been empirically validated [10], [15], [24], [25]: 1) Random Forest [26] is a classifier containing multiple decision trees. It can be used to build binary or multi-classification prediction models with high accuracy, robustness, and stability; 2) Decision Tree is a model that uses a tree-like data structure to represent decision rules and classification results. It can quickly find the characteristic variables that distinguish different categories and is suitable for multi-classification prediction; 3) k-Nearest Neighbor (KNN) is a method that classifies instances into K classes with high accuracy. It is one of the simplest machine learning algorithms and is suitable for multi-classification prediction.

In this paper, we used the Random Forest, Decision Tree (J48), and KNN classifiers that come with the Weka toolkit to predict the severity of defects introduced by changes. We directly applied the default setting of each classifier in Weka for our prediction models.

Model Training and validation. Numerous studies of JIT defect prediction have used the ten-fold cross-validation method

![Image](https://www.cs.waikato.ac.nz/ml/weka/)
to train and validate prediction models [18], [27]. The ten-fold cross-validation method first disrupts the dataset and divides it into ten parts. Then it selects one of them as the test set and the remaining nine as the training set. A total of ten experiments are run, and the average of the ten results is taken as the evaluation result of the ten-fold cross-validation. The advantage of this method is that each set of data can be used as training data and test data, so as to avoid over-learning or under-learning.

D. Model Evaluation

In this paper, well-known precision, recall, F1-measure, and AUC are selected to evaluate the prediction models’ performance. Precision and recall are the indicators of evaluation models commonly used in machine learning. Precision represents the proportion of all defects whose severity is correctly classified as \( i \) (i.e., 0, 1, 2, 3, or 4) to all defects classified as \( i \); recall refers to the proportion of all defects whose severity is correctly classified as \( i \) to all defects whose severity is \( i \). To harmonize and average precision and recall, we also used the metric of F1-measure, which is commonly used to measure the accuracy of JIT defect prediction models [8], [10], [27]. In a project, the labeled data is often imbalanced, that is, defect-introducing changes with different severity don’t distribute equally. Thus we added AUC into the evaluation metric suite. AUC is often used to evaluate the predictive ability of models built with unbalanced data [8], [23]. AUC is the Area Under the Curve of the receiver operating characteristic. Its scope is in \([0,1]\). The higher the value of AUC, the better the prediction performance of a model.

V. Evaluation

A. Subjects

We selected ten open-source projects from Apache \(^4\) as our subjects. The number of commits for each project ranges from 2,414 to 16,025 before May 6, 2022. Each severity level of the projects we selected contains at least 30 defect-introducing changes. We set this standard to guarantee the non-triviality of our selected subjects, and the dataset used for analyses is large enough to substantiate the reliability of the results.

Table II reports the statistics of defect-introducing changes at different severity levels: #1 (i.e. Minor or Trivial), #2 (i.e. Major), #3 (i.e. Critical), and #4 (i.e. Blocker). The first row shows the number of changes that introduced defects at #1 severity level (i.e. Minor or Trivial), and the following line shows the proportion of it to all defect-introducing changes. Similarly, rows 3 - 8 present the number of changes at #2, #3, #4 severity level and their proportion, respectively. Row 9 represents the proportion of the sum of the changes at #2, #3, #4 severity levels to all defect-introducing changes. Row 10 shows the total number of commits. The last two rows show the total number of defect-introducing commits and the proportion of it to all studied commits.

\(^4\)https://apache.org/

From this table, we can make a few straightforward observations: 1) a relatively large portion (from 19.5% to 46.4%, 30.0% on average) of all commits have induced defects, meaning that it is necessary to predict potential defects in advance when developers submit code changes; 2) The numbers of defect-introducing changes at different severity levels are various. The majority of defect-introducing changes are at the severity level #2, the percentages range from 29.1% to 78.5% (60.9% on average). The second largest proportion is at the severity level #4, reaching an average of 16.4%, and the third one is at the severity level #3, with an average of 13.0%. Defect-introducing changes at the severity level #1 occupy the smallest proportion with an average of 9.7%. According to the interpretation in prior work [28], Blocker, Critical, and Major represent high severity levels, while Minor and Trivial represent low severity. The above results show that the changes that introduce high-severity defects account for a really large proportion of all defect-introducing changes (from 80.8% to 97.9%, 90.3% on average). Because most of the high severity defects need to be fixed as soon as possible to avoid serious impact on the software system, it is necessary to predict the severity of the introduced defects just in time, which will help the development team to fix more serious defects in the shortest possible time.

B. Research Questions

RQ: What is the performance of our models on JIT defect severity prediction?

This question will show the predictive ability of our models, that is, whether the defect-introducing changes to be submitted by developers can be effectively predicted just in time. Since most serious defects need to be repaired as soon as possible and the quality of defect repair should be guaranteed to avoid serious impact on the software system, the timeliness of feedback defect severity should be fast and the precision of defect location should be high. At the same time, it is necessary to allocate defect repair personnel reasonably and avoid introducing new defects in the process of defect repair as far as possible. Therefore, it is very necessary to predict the severity of defects just in time, which can improve the efficiency and quality of defect repair and make greater use of limited resources.

C. Results

In each project, the amount of data with the defect severity level of 0 is far greater than that with the defect severity levels of 1, 2, 3, and 4. Therefore, this paper samples the data with the defect severity level of 0 in each project, so that its data amount is the same as the average data amount of the remaining levels. We used four metrics of precision, recall, F1-measure, and AUC to evaluate the predictive ability of our models. Tables III, IV, and V show the experimental results when using Random Forest, Decision Tree, and KNN classifiers, respectively.

We can find that the average precision of the models built by Random Forest is 0.552, the average recall is 0.579, the
average F1-measure is 0.528, and the average AUC value is 0.729; For the models applying the Decision Tree algorithm, the average precision is 0.479, the average recall rate is 0.494, the average F1-measure is 0.485, and the average AUC value is 0.619. For the models applying the KNN algorithm, they have an average precision of 0.466, an average recall rate of 0.468, an average F1-measure of 0.467, and an average AUC value of 0.593.

The above results have demonstrated that our approach could effectively conduct JIT defect severity prediction. In addition, in terms of the development languages of the projects, because our data labeling work is based on the tool SZZ Unleashed (it is suitable for multiple development languages), our models are suitable for predicting projects in multiple development languages. From Tables III, IV, and V, we can also find that our models have a good predictive ability for projects with Java, Python, or C++ as the main development language. The number of commits of the projects studied in this paper ranges from 2,414 to 16,025, the span is relatively large. Whether it is small sample data or large sample data, our models still perform well, and we can also find that the more the number of project commits, the better the prediction ability of the models. Therefore, the prediction results of our models are credible, and the scope of application of the models is relatively wide.

### VI. Discussion

In this paper, we only studied ten open-source projects, thus we cannot guarantee that the experimental results are generalizable to all projects, and there may exist some extra discoveries that have not yet been exhibited in our paper. But we have partially addressed this problem by selecting projects with different sizes, ages, and domains.

We used the ten-fold cross-verification method for our models’ training and validation. Although some studies [8], [18], [27] and our research have achieved good prediction results, Tan et al. [29] pointed out that the cross-validation method divides the data set randomly, which may take the future code change information as the training set and the past code change information as the test set. Therefore, in our future work, we plan to adopt some other methods, such as the time-wise-cross-validation method [9], out-of-sample bootstrap technique [11], or cross-project-validation method [9] in our approach.
In this paper, we build the JIT defect severity prediction models by using Random Forest, Decision Tree, and KNN Classifiers, respectively. Through the evaluation of ten open-source projects, we have presented that our derived models can effectively predict defect severity just in time. More specifically, the prediction model using Random Forest can get a precision of 0.552 on average, and the average recall, F1-measure, and AUC are 0.579, 0.528, and 0.729, respectively. For the prediction models using Decision Tree, the average values of precision, recall, F1-measure, and AUC are 0.479, 0.494, 0.485, and 0.619, respectively. For the prediction models using KNN, the average precision, recall, F1-measure, and AUC are 0.466, 0.468, 0.467, and 0.593, respectively. In addition, we find the majority (90.3% on average) of defect-introducing changes are at a high severity level.

The rational allocation and efficient utilization of resources are particularly important in the process of defect repair. Thus it is necessary to predict the severity of potential defects when the code is committed. We believe that with the help of our prediction models, the development teams can make more effective use of limited resources and improve the quality and efficiency of defect repair.

Numerical results show that the prediction model using Random Forest can yield better performance than those of the other two models. The average precision, recall, F1-score, and AUC of the prediction model using Random Forest are 0.586, 0.593, 0.590, and 0.729, respectively. For the prediction models using Decision Tree, the average precision, recall, F1-score, and AUC are 0.485, 0.498, 0.487, and 0.669, respectively. For the prediction models using KNN, the average precision, recall, F1-score, and AUC are 0.462, 0.463, 0.462, and 0.593, respectively. In addition, we find the majority (90.3% on average) of defect-introducing changes are at a high severity level.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under grant No. 62002129, The Knowledge Innovation Program of Wuhan-Shuguang Project under grant No. 2022010801020280.

REFERENCES

Abstract—Numerous software defect prediction methods utilize semantic information and software metrics as code features, neglecting the structural knowledge inherent in the source code. Other studies improve feature completeness by simply combining different types of defect indicators, which causes information redundancy. To address these challenges, this paper proposes a novel software defect prediction method that incorporates multi-type features and performs feature selection. Firstly, semantic and structural features are extracted by Text Convolutional Neural Network (TextCNN) and Graph Isomorphism Network (GIN) from Abstract Syntax Tree (AST) and Program Dependency Graph (PDG), respectively, which are combined with software metrics to build a multi-type feature set. Then, Recursive Feature Elimination with Cross-Validation (RFECV) integrating a novel feature importance measure is utilized to remove redundant features and generate a feature subset. Finally, a prediction model for classification is established based on the feature subset. The experiments validated the effectiveness of multi-type features and the improved RFECV. Overall our proposed method outperforms state-of-the-art techniques on nine Java open-source projects.

Keywords—software defect prediction; feature extraction; feature selection; recursive feature elimination

I. INTRODUCTION

To enhance software quality and reduce testing costs, software defect prediction (SDP) technology has emerged as a research hotspot in software engineering and software reliability assurance in recent years. The software defect prediction model is undertaken to identify software classes or modules that are more likely to fail and enable the minimization of maintenance costs before product deployment. Furthermore, this technique can provide a reference for testers to help them quickly grasp the overall quality of the software, and allocate test resources more reasonably.

Current SDP methods [1–3], extracting features from source code, tend to emphasize software metrics and semantic features, resulting in the lack of comprehensive information, especially structural information. Some studies [4] have attempted to improve software defect prediction by expanding the feature set and extracting features from various code abstract structures. However, the increase in the number of features does not necessarily lead to better prediction performance. Expanding the feature set may result in the inclusion of redundant information, which negatively impacts the prediction performance.

To solve these problems, this paper proposes a novel software defect prediction model named TGR (TextCNN-GIN-RFECV). TGR utilizes TextCNN [5] and GIN [6] to extract semantic and structural features from source code, respectively. The comprehensive feature set is composed of the extracted features and software metrics. Then, we propose the improved RFECV [7] to eliminate redundant features and generate a feature subset. Finally, the selected feature subset is fed into a classifier for training and prediction. In summary, this paper makes the following contributions:

- In order to enrich the feature information, we introduce a novel multi-type feature representation that combines semantic, structural, and software metric features.
- We modify the feature importance measure in RFECV by considering the real category, correlations with other features, and the classifier coefficient. And we utilize the improved RFECV to generate an appropriate feature subset and reduce redundant information.
- We reveal the impact of multi-type features and the improved RFECV on the overall performance of TGR by conducting experiments on nine Java open-source projects.

II. RELATED WORK

Software metrics are essential tools for assessing the quality and performance of software products and development processes. These metrics can significantly influence the accuracy of SDP models, including the widely used lines of code (LOC) and C&K metrics [8]. In the early stage of research, some studies use software metrics and traditional machine learning algorithms to predict and estimate whether software module contains defects, such as Logistic regression (LR) and support vector machine (SVM) [9].

Abstract Syntax Tree (AST) is widely used in software defect prediction research. Deep learning methods are often used to extract semantic features from ASTs, such as Convolutional Neural Networks (CNNs) [1], Long Short-Term Memory (LSTM) [2] networks, and Deep Belief Networks (DBNs) [3]. In addition, Code Property Graph (CPG) [10], Class Dependency Network (CDN) [4], and other code representations are also employed for structural feature extraction. Uddin et al. [11] use the Bidirectional Encoder Representations
from Transformers (BERT) pre-training model to obtain the vector representation of each word in the source code, which is utilized to train bidirectional LSTM for SDP.

There are many studies trying to ensure the quality of feature sets through feature selection. Zhu et al. [12] combine the whale algorithm and the complementary simulated annealing algorithm to select features. Saifan et al. [13] assess the effectiveness of feature selection techniques, including Principal Component Analysis, Pearson’s correlation, Greedy Stepwise Forward selection, and Information Gain.

III. METHOD

In this section, the proposed method TGR is described in detail. The entire framework of TGR is given in Fig. 1, which is divided into two stages: the feature extraction stage and the feature selection stage.

A. Feature extraction

1) Parsing source code: ASTs are facilitated by javalang\(^1\) from source code files. The nodes and granularity specified by Wang et al. [3] is adopted to form ASTs. Each AST is traversed in pre-order to obtain a node sequence. The node sequence preserves the original semantic order of the code by pre-order traversal and is maintained at an appropriate length, avoiding the redundancy of the source code. These sequences incorporate the semantic information of the corresponding source code file in a more retrenched representation. Furthermore, a dictionary is established based on the node set to vectorize the node sequence, enabling TextCNN to process and analyze.

AST is adept at capturing semantic information but falls short in capturing structural information. Compared with semantic information, structural information has more complex representations and greater depths of features. In order to explore the abundant structural information contained in the source code, PDG is introduced into the prediction method. As an extension of Control Flow Graphs (CFGs), PDGs capture not only control dependencies but also data dependencies among the program statements. Their nodes represent a program statement, while their edges reflect the data transfer and program execution flow between nodes, which can more comprehensively exhibit the structural information of the program. This stage utilizes SourceDG [14] to generate PDGs from source code, as the source of structural features.

2) Extracting features: After obtaining the AST node vector and PDG to represent the semantic and structural information of the code, it is crucial to choose appropriate feature extraction models. Given the distinct feature depth and representation forms of semantic and structural information, they should be processed by different models. Thus, TextCNN and GIN are chosen to individually process node vectors and PDGs, and extract the semantic and structural features.

The network structure of TextCNN comprises a word embedding layer, a convolutional layer, and a pooling layer. The convolution layer employs three different sizes of convolution kernels to help the node comprehend its context in the code at different scales. Every kernel transforms the input vector to a feature map as the input of the pooling layer, and the outputs of the pooling layer are then connected. Subsequently, two fully connected layers with different output sizes are employed. The first fully connected layer generates predictions for calculating loss and performing backpropagation. When the training is completed, the output of the second fully connected layer is taken as the semantic feature of TGR.

Noted that, unlike LSTM and other neural network models, the output length of the TextCNN used in TGR is solely determined by the model parameters, and is independent of the length of the node vector. Each element of the output semantic feature is derived from the same network structure and represents the entire input sequence, which facilitates the feature selection stage of TGR.

GIN, a variant of Graph Neural Network (GNN), enhances performance on graph classification and graph embedding tasks. GIN utilizes two main operations, namely graph convolution and graph readout, to update node features and generate graph-level representations, respectively. During graph convolution, messages are passed on the edges of a PDG and update the node features to hidden representations. The hidden representations are generated through multiple message passes and subsequently passed to the graph readout part. During the graph readout, we choose summation to aggregate and summarize node features to generate a graph-level representation. The resultant graph-level representations are then passed into fully connected layers, and the outputs are summed as the structural features of TGR. Besides, the structural feature is used as input to another fully connected layer to generate predictions for calculating losses and performing backpropagation.

Since software metrics are crucial features in defect prediction models. The third type of feature in TGR, metric features, are concatenated with semantic and structural features to create a possibly abundant and comprehensive feature set after the three types of features are z-score standardized separately. The utilization of multi-type features enables the classifier to predict with the more comprehensive feature set.

B. Feature selection

Since a large number of features are extracted in the feature extraction stage, potentially redundant and irrelevant information is introduced into the prediction model. To address the redundancy, a feature selection stage is incorporated to acquire a more beneficial feature subset.

RFECV selects features considering the classifier performance. The Recursive Feature Elimination (RFE) algorithm iteratively trains a classifier and eliminates several features with low classifier coefficients until reaching the preset minimum number of features. RFECV uses RFE to get the feature ranking, then selects feature subsets of different sizes based on the ranking to perform cross-validate (CV). The CV can automatically determine the size of the feature subset, and avoid the adverse effects of the unreasonable setting of the feature subset size as a hyperparameter. Finally, the subset

\(^1\)https://github.com/c2nes/javalang
with the highest average score is chosen as the optimal feature subset.

The feature eliminated in each recursion of RFE is determined by the coefficients of the classifier, in other words, the feature importance of RFE is measured by the classifier coefficients. These coefficients indicate the feature’s impact on the classification results output by the classifier. However, the output of the classifier is not entirely correct, resulting in wrongly assigning importance to features that are not related to the real category. Moreover, feature importance is not only reflected in the coefficients, but also in factors such as correlations with other features, and correlations with real categories. Therefore, the feature importance measure in RFE is replaced by our defined comprehensive feature importance measure, which enables RFECV to select a feature subset more comprehensive, informative, and effective for classification.

In this stage, two coefficients are proposed to calculate the comprehensive feature importance with the classifier coefficients: category correlation coefficients and feature correlation coefficients. The category correlation coefficient complements the influence of the real category on feature importance. The Normalized Mutual Information (NMI) is used to measure the correlation between a feature and the real category vector after discretizing the feature. Suppose the number of features is N, \( x_i \) represents the \( i \)-th feature in the feature set, and \( y \) represents the real category vector. The formula for calculating the category correlation coefficient is given as:

\[
cc_i = \frac{2 \cdot (H(x_i) - H(x_i | y))}{H(x_i) + H(y)},
\]

where \( H(x_i) \) is the entropy of \( x_i \), \( H(x_i | y) \) is the joint entropy of \( x_i \) and \( y \), and \( cc_i \) is the category correlation coefficient of \( x_i \).

Feature selection often results in the removal of numerous features, causing the loss of valuable information contained in these features. With a limited number of remaining features, it is expected to select features that contain more comprehensive information. In this stage, features are categorized into three classes according to their type, namely, semantic features, structural features, and metric features. The feature correlation coefficient is calculated by the average of the correlations of a feature with all other features in the same class, which quantifies the amount of comprehensive information contained in the feature within its class. The correlation between features is measured by Pearson’s coefficient. The formula for calculating the feature correlation coefficient is as follows:

\[
fc_i = \frac{\sum_{j=1}^{N} i \cdot r(x_i, x_j)}{N},
\]

\[
r(x_i, x_j) = \frac{\sum_{k=0}^{N-1} (x_i - \bar{x})(x_j - \bar{x})}{\sqrt{\sum_{k=0}^{N-1} (x_i - \bar{x})^2 \sqrt{\sum_{k=0}^{N-1} (x_j - \bar{x})^2}}
\]

where \( L_i \) and \( R_i \) are the index of the start and end of the feature class to which \( x_i \) belongs in the feature set, \( r(x_i, x_j) \) is the absolute value of Pearson’s coefficient of \( x_i \) and \( x_j \), and \( fc_i \) is the feature correlation coefficient of \( x_i \).

As shown in (3), the formula for the overall feature importance measure is a weighted sum of the classifier coefficient, category correlation coefficient, and feature correlation coefficient. The weights are denoted as \( w_1 \), \( w_2 \), and \( w_3 \). The classifier coefficient of \( x_i \) is \( coef_j \).

\[
score_i = w_1 \cdot |coef_j| + w_2 \cdot cc_i + w_3 \cdot fc_i
\]

\( score_i \) in (3) represents the comprehensive feature importance of \( x_i \), which is calculated based on the three factors. The higher the \( score_i \), the more important \( x_i \) is. The absolute value of classifier coefficient \( |coef_j| \) in (3) reflects the importance of \( x_i \) to the classifier output, while the category correlation coefficient \( cc_i \) represents the correlation between \( x_i \) and the real categories, and the feature correlation coefficient \( fc_i \) evaluates the comprehensiveness of \( x_i \) in terms of the information it carries. By incorporating these factors, our proposed comprehensive feature importance measure provides more comprehensive and instrumental information for RFECV, which can ultimately enhance the classification performance of the classifier.

Finally, the feature subset is fed into the same type of classifier as used in RFECV to train and predict source files are defective or not. The classifier used in TGR is LR.

---

**Figure 1.** The entire framework of TGR method
IV. Experiment Setup

A. Datasets

The experiments in this paper are constructed on nine Java open-source projects from the PROMISE repository [15], as presented in Table I. The older version of projects is utilized as the training set, and the newer version is the test set.

<table>
<thead>
<tr>
<th>Project</th>
<th>Releases</th>
<th>Avg. Files</th>
<th>Avg. Defect rate(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ant</td>
<td>1.5 1.6</td>
<td>322</td>
<td>24.3</td>
</tr>
<tr>
<td>camel</td>
<td>1.4 1.6</td>
<td>919</td>
<td>18.1</td>
</tr>
<tr>
<td>jedit</td>
<td>4.0 4.1</td>
<td>309</td>
<td>25.0</td>
</tr>
<tr>
<td>log4j</td>
<td>1.0 1.1</td>
<td>244</td>
<td>62.0</td>
</tr>
<tr>
<td>lucene</td>
<td>2.0 2.2</td>
<td>221</td>
<td>53.2</td>
</tr>
<tr>
<td>poi</td>
<td>2.5 3.0</td>
<td>827</td>
<td>64.0</td>
</tr>
<tr>
<td>synapse</td>
<td>1.0 1.2</td>
<td>202</td>
<td>23.0</td>
</tr>
<tr>
<td>velocity</td>
<td>1.5 1.6</td>
<td>443</td>
<td>49.7</td>
</tr>
<tr>
<td>xalan</td>
<td>2.5 2.6</td>
<td>844</td>
<td>47.3</td>
</tr>
</tbody>
</table>

B. Evaluation

In order to comprehensively evaluate the experimental results, Area Under Curve (AUC) and Matthews correlation coefficient (MCC) are adopted in this paper. AUC is defined as the area enclosed by the receiver operating characteristic (ROC) curve and the coordinate axis. The horizontal axis of the ROC curve is the False Positive Rate (FPR), and the vertical axis is the True Positive Rate (TPR). The FPR and TPR can be calculated by (4). MCC is an evaluation indicator comprehensively considering the four basic evaluation indicators in the confusion matrix: TP, TN, FP, and FN, and is calculated by (5).

\[
\begin{align*}
FPR & = \frac{FP}{TP + FN} \\
TPR & = \frac{TP}{TP + FN} \\
MCC & = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}
\end{align*}
\]

C. Parameters Settings

The PROMISE repository not only provides the number of defects of source files but also provides 20 software metrics used in TGR, including Weighted Methods per Class (WMC), LOC, etc. The dimensionalities of semantic features and structural features are set to 50 and 30. The kernel sizes of TextCNN are 4, 5, and 6, and the number of every type of kernel is 100. The three weights of (3), \(w_1\), \(w_2\), and \(w_3\), are set to 0.6, 0.2, and 0.2, respectively. The RFECV utilizes 5-fold cross-validation, removes 1 feature at each iteration, uses the F-measure as the scorer of CV, and selects 20 features at least. The batch size and epoch of the experiment are respectively set to 32 and 100, and the experiments are repeated 10 times.

D. Experimental Design

In order to analyze and evaluate the proposed TGR method, three Research Questions (RQ) are posed:

**RQ1:** Do the proposed TGR method perform better than the baseline methods?

**RQ2:** How do the multi-type features, comprehensive feature importance measure, and the improved RFECV perform in TGR?

**RQ3:** Can the proposed method be effective for the prediction performance improvement of other SDP methods?

To answer RQ1, four baseline methods are set to compare with TGR.

1) LR: A traditional method using software metrics as features of LR classifier.
2) SVM: A traditional method using software metrics as features of SVM classifier.
3) DP-CNN: A deep learning method using a standard CNN to extract abstract semantic features from the pre-order sequences of ASTs, and using LR as a classifier with the extracted features and software metrics as features [1].
4) BiLSTM: A deep learning method using a Bi-directional LSTM automatically learns the semantic and contextual information from the program’s AST, choosing LR as the final classifier [2].

The ablation experiment is to explore the influence of a part of a model by deleting this part and testing the performance of the model. Therefore, in order to answer RQ2, the following methods are set up:

1) TextCNN: A method using TextCNN to extract semantic features from the pre-order sequence of ASTs, which has the same TextCNN as the one in the TGR method, and using LR as a classifier.
2) TG: A method without feature selection, using the same multi-type features, feature extraction stage, and classifier as in TGR.
3) TG+: A method using the same feature extraction stage and classifier as in TGR, and using standard RFECV (a RFECV using classifier coefficients as feature importance measure) to select features. The parameters of RFECV are consistent with those in TGR.

In order to compare the above methods accurately and fairly, the features used in these methods are limited to the features obtained from the same experiment. In detail, after feature extraction in each experiment, the extracted features are input into different classifiers after different processing, corresponding to TextCNN, TG, TG+, and TGR methods.

Finally, to verify the effect of the multi-type features and feature selection proposed in this paper on other classifiers, SVM classifiers are employed to replace all the LR classifiers in TGR, named TGR-SVM. This experiment serves to answer RQ3.

Besides, due to the class imbalance of the dataset, oversampling is used in all methods mentioned in this paper at the data preprocessing stage.

V. Experimental Results

**RQ1:** Do the proposed TGR method perform better than the baseline methods?

Table II shows the average AUC values and MCC values of TGR and other baseline methods on each project. TGR's
average AUC value and MCC value are 0.689 and 0.353, respectively. It is noteworthy that TGR outperforms the baseline methods in the majority of projects, and that its average AUC and MCC values are superior to those of all baseline methods. Compared with LR and SVM, which use software metrics as features, the AUC values of the TGR method achieve improvements of 4.9% and 5.2%, respectively. Compared with deep learning methods, DP-CNN, and BiLSTM, the AUC values of the TGR method achieve improvements of 12.4% and 13.7%, respectively. Especially, the biggest improvement of TGR’s AUC is 36.1% on the ant project compared with BiLSTM. The experimental results demonstrate that TGR overall performs best compared with all baseline models.

**RQ2: How do the multi-type features, comprehensive feature importance measure, and the improved RFECV perform in TGR?**

Table III exhibits the average AUC and MCC values of TextCNN, TG, and TG+ on each project. To facilitate a more intuitive analysis of the impact of feature combination and feature selection stage in TGR, we present box plots depicting the AUC values of TextCNN, TG, TG+, and TGR on each project, which are displayed in Fig. 2. According to Table II and III, compared with TextCNN which utilizes only semantic features, TG, which uses multi-type features, demonstrates improvements in average AUC. Compared with TG, the proposed TGR with improved RFECV exhibits increases in average AUC. As Fig. 2 shows, it can be observed that the majority of the boxes representing TextCNN, TG, and TGR, display a gradual increase in position. These findings demonstrate that the multi-type feature and improved RFECV can improve the prediction performance of TGR.

**On the other hand,** the length of the box and whisker in box plots reflects the volatility of the data to some extent. According to Fig. 2, compared with TextCNN, the boxes of TG changes irregularly in length. It is worth noting that the boxes for TGR are the shortest among most projects. Hence, it is verified that the AUC values of the TGR’s prediction results exhibit a more concentrated distribution on most projects. These findings indicate the improved RFECV contributes to improvement in the stability of TGR.

To explore the impact of the proposed comprehensive feature importance measure in improved RFECV, Fig. 3 shows the average improvement of the MCC value of the TG+ and TGR methods relative to TG in each Project. It is intuitive that in five projects, the average improvement of TG+’s MCC is negative. In contrast, the average improvement of TGR’s MCC is positive in eight projects. These trends are also reflected in Fig. 2, where it can be observed that, apart from the log4j and synapse projects, the boxes of TG+ do not exhibit a significant elevation compared with those of TG. Moreover, the boxes of TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TG+. And The average AUC and MCC in Table III of TG+, are lower than those in TGR are almost higher compared with TGR.

**TABLE II. AUC and MCC FOR TGR VERSUS BASELINE METHOD**

<table>
<thead>
<tr>
<th>Project</th>
<th>TextCNN</th>
<th>TG</th>
<th>TG+</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>MCC</td>
<td>AUC</td>
<td>MCC</td>
</tr>
<tr>
<td>ant</td>
<td>0.719</td>
<td>0.366</td>
<td>0.715</td>
</tr>
<tr>
<td>camel</td>
<td>0.597</td>
<td>0.157</td>
<td>0.614</td>
</tr>
<tr>
<td>jedit</td>
<td>0.712</td>
<td>0.369</td>
<td>0.726</td>
</tr>
<tr>
<td>log4j</td>
<td>0.760</td>
<td>0.501</td>
<td>0.728</td>
</tr>
<tr>
<td>lucene</td>
<td>0.617</td>
<td>0.232</td>
<td>0.597</td>
</tr>
<tr>
<td>poi</td>
<td>0.706</td>
<td>0.397</td>
<td>0.664</td>
</tr>
<tr>
<td>velocity</td>
<td>0.625</td>
<td>0.238</td>
<td>0.575</td>
</tr>
<tr>
<td>xalan</td>
<td>0.541</td>
<td>0.081</td>
<td>0.644</td>
</tr>
<tr>
<td>synapse</td>
<td>0.638</td>
<td>0.263</td>
<td>0.638</td>
</tr>
</tbody>
</table>

Average 0.657 0.289 0.655 0.295

**TABLE III. AUC AND MCC FOR TextCNN, TG, TG+**

<table>
<thead>
<tr>
<th>Project</th>
<th>TextCNN</th>
<th>TG</th>
<th>TG+</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>MCC</td>
<td>AUC</td>
<td>MCC</td>
</tr>
<tr>
<td>ant</td>
<td>0.713</td>
<td>0.378</td>
<td>0.708</td>
</tr>
<tr>
<td>camel</td>
<td>0.635</td>
<td>0.223</td>
<td>0.642</td>
</tr>
<tr>
<td>jedit</td>
<td>0.709</td>
<td>0.353</td>
<td>0.717</td>
</tr>
<tr>
<td>log4j</td>
<td>0.761</td>
<td>0.504</td>
<td>0.771</td>
</tr>
<tr>
<td>lucene</td>
<td>0.618</td>
<td>0.231</td>
<td>0.626</td>
</tr>
<tr>
<td>poi</td>
<td>0.644</td>
<td>0.275</td>
<td>0.649</td>
</tr>
<tr>
<td>velocity</td>
<td>0.648</td>
<td>0.284</td>
<td>0.663</td>
</tr>
<tr>
<td>xalan</td>
<td>0.642</td>
<td>0.284</td>
<td>0.661</td>
</tr>
<tr>
<td>synapse</td>
<td>0.672</td>
<td>0.341</td>
<td>0.675</td>
</tr>
</tbody>
</table>

Average 0.671 0.319 0.679 0.335

On the other hand, the length of the box and whisker in box plots reflects the volatility of the data to some extent. According to Fig. 2, compared with TextCNN, the boxes of TG changes irregularly in length. It is worth noting that the boxes for TGR are the shortest among most projects. Hence, it is verified that the AUC values of the TGR’s prediction results exhibit a more concentrated distribution on most projects. These findings indicate the improved RFECV contributes to improvement in the stability of TGR.
The average AUC values of TGR-SVM and SVM on each project are shown in Fig. 4, and the two dashed lines represent the average AUC values on all projects of the two methods. TGR-SVM exhibits superior performance over SVM, as evidenced by eight individual projects demonstrating an average AUC value higher than that of SVM. Notably, the highest improvement in the AUC value achieved by TGR-SVM is 0.089 on the velocity project. Moreover, TGR-SVM demonstrates a significant increase in its average AUC value across all projects compared with SVM. Seven AUC values of TGR-SVM exceed the average AUC value across all projects of SVM. The results demonstrate that the multi-type features and the improved RFECV in TGR are not confined to the LR classifier. Applying them to SVM classifiers also achieves increased prediction performance.

VI. CONCLUSION

This paper proposes TGR that comprises two main stages: feature extraction and feature selection. In the feature extraction stage, the semantic features and structural features are extracted from source codes, respectively. These features, along with software metrics, are concatenated to build a multi-type feature set. The feature selection stage employs an improved RFECV to eliminate the redundancy features, which is implemented by integrating the category correlation coefficients, the feature correlation coefficients, and the classifier coefficients as the feature importance measure. The selected feature subset is fed to the classifier to get predictions. Experimental results demonstrate that TGR outperforms the four baseline methods in prediction performance. The ablation experiments are conducted to validate the effectiveness of multi-type features and the feature selection stage in TGR. Future work will focus on exploring better methods for extracting more comprehensive defect features, and will apply TGR for cross-project software defect prediction.

VII. ACKNOWLEDGMENT

This work was supported by the second batch of cultivation projects of Pazhou Laboratory in 2022, No. PZL2022KF0008, and by the Zhongshan Produce and Research Fund, PR China under grant no. 210602103890051.

REFERENCES

Identifying Risks and Risk-Mitigation Strategies for Collaborative Systems during Requirements Engineering: A Goal-Oriented Approach

Kirthy Kolluri, Lawrence Chung
Department of Computer Science
The University of Texas at Dallas
Richardson, TX
{kirthy.kolluri, chung}@utdallas.edu

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

Abstract—If risks are not identified, they are unlikely to be addressed, possibly resulting in undesirable consequences, such as fatal accidents or even loss of human lives. Furthermore, risks should be considered not only in terms of system behavior but also events occurring in the system environment—i.e., in a collaborative setting in which the system and its environment work together towards certain goals the system is intended to help achieve. In this paper, we present a goal-oriented risk analysis framework, Murphy+G, in which non-functional requirements (hereafter, NFRs) are treated as softgoals to be achieved and systematically addressed in terms of both the system and its environment during requirements engineering, by adopting what is called the Reference Model. A study of a smartphone app, Theia, which is intended to help blind people navigate indoors, is used for the purpose of both illustration and experimentation. In this study, NFRs (e.g., safety, reliability, timeliness, etc.) are treated as softgoals, and risks (e.g., fall down, injury, etc.) are identified, along with risk-mitigation strategies for both the system and its environment, with the help of an activity-oriented ontology. To see both the strengths and weaknesses of Murphy+G, a systematic methodology for risk analysis for collaborative systems, a controlled experiment has been carried out, in terms of three different versions of Theia implementations. Feedback from students show improvements on the accuracy of the risk analysis and the risk mitigation strategies devised, as well as enhanced users’ experience with respect to increased confidence in navigating indoors, in a safe, timely, and reliable manner.

Keywords- risks, risk analysis, risk-mitigation, non-functional requirements (NFRs), softgoals, Reference Model (WRSPM Model), ontology

I. INTRODUCTION

The statement “A Risk unidentified is a Risk unaddressed” expresses the importance of risk identification, analysis and addressing the risks that are identified. Risk, which is defined as a situation or event where something of human value (including humans themselves) can be put at stake and where the outcome is uncertain”[14], is a phenomenon faced or caused by a person. In general, each action performed may have one to many risks associated with them which can range from a normal risk (e.g., missing route) to a serious risk (e.g., loss of life). The usage of smartphone apps (Collaborative Systems) has been on the rise and studies show that people spend an average of 3 hours and 10 1 minutes using their phone out of which 2 hours and 51 minutes is being spent on smartphone apps, hence many people are inclined towards using smartphone apps for day-to-day activities [20]. The Reference Model emphasizes that the requirements are satisfied by the collaboration between the user and the events in its environment, hence the term collaborative system. A previous study describes the importance of comprehensively identifying the key NFRs (Non-Functional Requirements) and appropriately handling them [10]. For example, with apps such as Theia, a smartphone app for helping blind people navigate indoors, there is always chance for the user (blind person) getting hurt, if risks, which are related to user goals, e.g., safety, reliability, timeliness, etc. (the so-called NFR softgoals [11]) are not identified or not appropriately handled. Lack of systematic methodologies to identify the most important NFR softgoals related to a requirement, the risks associated with them, and ways to mitigate them can lead to mishaps and undesirable things from happening.

We extend our previous research, the Murphy framework [12] [13], with goal-orientation. This new framework, the Murphy+G supports goal-oriented risk identification and analysis for collaborative systems. Two technical contributions have been made in this paper, including 1) a goal-oriented, activity-driven ontology for the Murphy+G framework which captures the most important goals, actions and risks related to the smartphone app (Theia), and 2) a technique to combine the Reference Model with the NFR framework for identifying the most important ways to achieve NFR softgoals (called operationalizing softgoals) in the context of the Reference Model, where requirements are met in terms of specification and domain, and choosing the most feasible risk-mitigation strategies.

The proposed approach is illustrated using a smartphone application, Theia, for helping blind people navigate indoors. A scenario using Theia, for helping blind people navigate indoors is used as the running example all through this paper for easy understandability. Stevie is a blind student who wants to attend a class in room 3.415. He uses a smartphone application to
navigate from his current location to his class, room 3.415. He
uses voice instruction to provide his destination to Theia. To
navigate to his destination Stevie must walk 10 steps forward
and turn right but we will elaborate only on the walking 10 steps
forward part for performing goal-oriented risk analysis. The rest
of this paper is structured as follows. Section 2 discusses related
work and adopted models for this work. Section 3 presents the
Murphy+G framework. Section 4 describes the experiments
conducted and their results. Section 5 discusses our observations
and threats to validity. Finally, Section 6 summarizes the paper
and future work.

II. RELATED WORK AND ADOPTED MODELS

We believe that this paper is one of the first to extend the
NFR Framework with the Reference Model and an activity-
oriented ontology, which corresponds to it. There has been
some work on goal-oriented risk analysis (e.g., see [1, 3, 6, 7,
11]) in literature. Our work is similar especially to [1] [8], in
the sense that it provides a qualitative goal-oriented risk analysis,
with an ontology, but, among other things, without adopting the
Reference Model or an activity-oriented ontology.

For this work we have adopted various models for
representing functional requirements, specification and domain
assumptions, non-functional requirements (softgoals), and
models to represent risks for which risk-mitigation strategies
are chosen to help avoid those risks.

A. For Requirements, Specification and Domain
Assumptions:

A picture of the Reference Model is shown in fig. 1, which
consists of Environment (E) and System (S). The environment
consists of Domain Assumptions (W or D) and Requirements
(R) and System consists of Program (P) and Machine (M) and
the Specification (S) is present in the intersection of E and S.
The environment has environment events eih and ev, and the
system has system events sh, sv and sh (hidden h) and visible (v)
events respectively. The user is a designator with an intention,
designated to perform any kind of actions in the environment.
The Reference Model [4, 5] 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 those kinds of
systems.

B. For Non-Functional Requirements:

For representing NFRs, there are several goal-oriented
frameworks, including KAOS [15], i* [16], and the NFR
framework [11], each with its own significance, quality, and
features. Adopting the NFR Framework [11] and other similar
works, we treat NFRs, such as safety, reliability, timeliness,
etc., as softgoals, which convey the sense that they typically
have no clear-cut definition or absolute criteria to determine
their satisfaction. Hence, instead of logical “satisfy”, we use
“satisfice” with different degrees of contribution. Each softgoal
can be either AND or OR decomposed into sub-softgoals or
contribute towards satisfying another softgoal either fully or
partially positively (MAKE or HELP), or fully or partially
negatively (BREAK or HURT). A label propagation
mechanism evaluates the effect of a decision on upper softgoals,
with a label - Satisfied, Denied, Conflict, or Undetermined.
Softgoals and relationships between softgoals are visualized by
using Softgoal Interdependency Graph (SIG) (see Fig. 3 for
examples).

Operationalizing softgoals are concrete functional
requirements (FRs) (risk-mitigation strategies in this paper),
which are implemented as features in the projected software
system to address various risks (e.g., In Theia, to address the
risk “Incorrect number of steps taken”, risk-mitigation
strategies could be any features that provide help with counting
steps), which can satisfice the NFR softgoals (e.g., user goals
such as safety, reliability, etc.).

C. For Risks:

There are various models for problem (risk in this paper)
representation and root cause analysis, including Fish Bone
Diagram [22], Problem Interdependency Graph (PIG) [19],
Fault Tree Analysis (FTA) [18], Cause Effect Graph [20],
Failure Mode and Effect Analysis (FMEA) [17]. While FTA, is
used when information is available about AND/OR logical
relationships among root causes, Fish Bone Diagram is used
when the problem is about relationships among root causes.
Causal Mechanism graph help design better assessment models
and risk control approaches since causal mechanisms capture
the causal entities that determine software dependability, while
FMEA reviews components and subsystems to identify
potential failure modes in a system and their causes and effects,
while providing a scale to understand the likelihood of failure,
while our risk analysis uses an activity-oriented ontology to
systematically identify risks.

We adopt and use PIG, since it not only resembles a SIG,
but also since problems (potential problems) are similar to risks.
The problems (risks in this work) and relationships between
problems are represented using a Problem Interdependency
graph. Problems represent potential problems until they are
realized using label propagation, hence we consider them as
risks for this work.

III. MURPHY+G: A GOAL-ORIENTED FRAMEWORK FOR
UTILIZING GOALS AND REFERENCE MODEL FOR RISK
ANALYSIS

Fig. 1: Reference Model showing Domain (D or W), Requirement (R),
Specification (S), Program (P) and Machine (M) along with their events

245
Murphy+G framework offers an activity-oriented ontology. The framework then introduces goals related to requirements by extending NFR framework with the Reference Model. Additionally, the framework offers (SIG-PIG) + Reference Model (RM) Graph which represents the process graphically, thereby bringing in the notion of Reference Model to SIG-PIG to analyze and identify Specification and Domain Assumptions to satisfy the Requirement (FR).

A. Ontology of the Murphy+G Framework:

Fig. 2 shows a high-level domain independent ontology which includes some concepts that we have discussed in section II. Ontology is meant to refer to the categories of essential individual concepts, relationship between the individual concepts and constraints on individual concepts and on the relationships between individual concepts, as in [9]. It depicts salient concepts such as actions, agents, risks, goals, concepts related to the Reference Model (requirements, specification, etc.).

An agent is someone/something that has the capability to perform an action, oftentimes with some intention (goal, in this paper), which might be attributed to by another agent. Our ontology recognizes people, software, and hardware as three types of agents. As for goals, we consider softgoals; softgoals can be an operationalizing softgoal (a feature that can be implemented in the system to be developed; e.g., screen tapping mechanism – where the user taps the screen for every step taken, etc.). Risks are potential problems that the agent (Person) may face, and a risk hinders the achievement of a goal. Hence, we come up with risk mitigation strategies which help achieve the user’s goals by alleviating some risks faced by the user.

B. Process offered by the Murphy+g Framework:

The Murphy+G process consists of 5 steps:

Step 1: Capture most important NFR Softgoals. In this step, most important users’ goals are explicitly captured as softgoals from the stakeholders by using a questionnaire. The questionnaire is not shown here due to space limitation. Based on the answers provided by the users’ of the app, the most important NFRs (Safety, Timeliness, Reliability, etc.) for this app are captured.

Since most of the users’ (e.g., blind person) express their goals as being safe (Safety), have an application and smartphone that they can rely on (Reliability) (e.g., smartphone app, smartphone, etc.) and being able to reach their destination on time (Timeliness), we have capture these softgoals as the most important at the highest level for this work. These softgoals are represented using a cloud symbol, as shown in step 1 in fig. 3.

Step 2: Decompose softgoals into sub-softgoals and identify risks associated with the softgoals. As discussed in step 1, Safety, Reliability and Timeliness are captured as the most important softgoals at the highest level. Due to space restriction, we will discuss only safety softgoal here. Safety is the type of the softgoal, and User is the topic as discussed in [11]. A softgoal can be decomposed either using type or topic. Firstly, Safety[User] is refined using an AND decomposition by topic, i.e., Safety[User] is decomposed into Safety[Blind Person] and Safety[Passer By] sub-goals, where the former is further decomposed by type into Attentiveness[Blind Person] and Committing Less Errors[Blind Person].

Step 3: Identify potential problems associated with sub-softgoals. For the low-level sub-softgoals identified in Step 2, which are Attentiveness[Blind Person] and Committing Less Errors[Blind Person], risks that could impede with achieving the goals are identified in step 3. The risks relevant to the goals are identified, which may be further decomposed into sub-risks (if applicable), until they are at a basic level. To address these identified risks, some functional features (operationalizations), could be implemented in the projected software system, to help avoid the identified risks, as shown in step 3 in fig. 3. In this example, Wrong Action performed could break the Committing Less Errors[Blind Person] softgoal. Wrong Action Performed risk is further OR decomposed into Walk Incorrect Number of...
Steps. Turn in Wrong Direction or Take wrong Route sub-risks.

Step 4: Find operationalizing softgoals (FRs) for the risks. This step identifies the risk-mitigation mechanisms, which are the operationalizing softgoals or the functional requirements that may be implemented in the projected system.

For each of the leaf risks, an operationalization represented by the yellow cloud as shown in fig.3, helps alleviate that particular risk. As discussed in the example in step 3, we OR refine Wrong Action Performed into Walk Incorrect Number of Steps, Turn in Wrong Direction or Take wrong Route. Considering Walk Incorrect Number of Steps, o_r1: Phone Vibration, o_r2: Phone reading out the step number and o_r3: Screen Tap for every step taken can be some risk mitigation strategies that can help avoid “Walking wrong number of steps” risk. All these risk-mitigation strategies either hurt or break the risks, which satisfies the risk-mitigation strategy.

Step 5: Identify corresponding specification and domain assumptions. In this step, we bring in the specification and Domain assumptions that satisfy the requirements. The Reference Model, states that the specification (S) and the domain assumptions (D) together satisfy the requirement [4, 5].

\[ S, D \vdash h \]

Hence, every requirement must have a S and D that satisfies it. The requirement (risk-mitigation strategy) must be satisfied by S, D to help alleviate the risk. For a requirement to be satisfied, there must be some domain assumptions that must hold, and some system specification that needs to be implemented for a requirement (operationalization/ risk-mitigation strategy) to be satisfied. Using the example in Fig. 3, and as discussed in step 2.3, let us consider the risk-mitigation strategy o_r2: Phone reading out the step number, the domain assumptions d1 is that the hardware must work without any problems and d2 is that the volume is in audible range for the user to hear. The corresponding specification, s1 is that the instruction must be loud and clear.

After identifying specification and domain for the all the operationalizing softgoals, the best solution which helps avoid that particular risk must be selected to be implemented. For that, we perform a trade-off analysis to explore among alternatives to select the option that helps alleviate the problem.

C. Perform trade-off analysis for risk mitigation strategies:

In this step, all the risk mitigation strategies identified are considered and a trade-off analysis is performed to explore among alternatives. These risk-mitigation strategies are analyzed using various important NFR softgoals, and the operationalization/risk-mitigation strategy that helps avoid the risk in a better and an efficient way is selected during the requirements engineering phase and is implemented as a feature in the projected software system. For example, in Table. 1, we have operationalizing softgoals on one side of the table and softgoals on the other.

If we consider “Phone reads out step number” feature, it is given a contribution of ++ since the blind person is not involved in performing any action, hence the chance of a risk happening is less when compared to “screen tap for every step taken” which was given a +, which means that this feature is helpful partly (sometimes). If the user must tap the screen for every step taken, there may be a chance for some risks to arise such as blind person forgetting to tap the screen, screen tap not being recorded, etc.

Similarly, for the “Phone vibrating for every step taken”, there may be some risks such as the blind person may walk very fast, or the smartphone app may not be able to detect the steps, or the blind person may not be able to recognize the vibration, etc. Hence, with regards to Safety softgoal, Phone reading out the step number is the best risk-mitigation strategy that could be implemented. In a similar fashion, trade-off analysis was performed by taking all other softgoals into consideration. We have identified that “phone reading out the step number” while the person is walking as the best risk-mitigation strategy to be implemented to address the risk “Walking incorrect number of steps”.

Validation Tools: Murphy+G Assistant has been implemented to validate the risk identification and analysis process and to devise risk-mitigation strategies. We have adopted the Murphy Assistant tool from our previous work [12, 13] and extended it.
TABLE 1: An example table showing the trade-off analysis for selecting the best risk-mitigation strategy

<table>
<thead>
<tr>
<th>Risk-Mitigation Strategy</th>
<th>Safety (User)</th>
<th>Timeliness</th>
<th>Reliability</th>
<th>Easy to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen tap for every step taken</td>
<td>++</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Phone reads out step number</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Phone Vibrates for every step</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

with the concept of goals, which is a semi-automated risk analysis tool, where the user of the application has to setup the ontology, provide a requirement and the most important softgoals before performing risk identification and analysis. Fig. 4 shows an example snapshot of the validation tools in action. 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. We also developed two versions of Theia, with the risk-mitigation strategies devised using the results from the Murphy+G Assistant. The underlying code and the snapshots of the tool in action can be found at https://github.com/indoornavigation0/Murphy.git.

IV. EXPERIMENTATION AND VALIDATION

The authors of the paper has conducted various controlled experiments, while also involving feedback from students (more on this in the next section), to see the strengths and weaknesses of the Murphy+G framework, using the results from the Murphy+G tool, that the authors experimented with. The risk-mitigation strategies from the tool have been considered and three different implemented versions of the smartphone app (Theia) were used for experimentation.

Experimental setup:

We have developed three versions of Theia additionally, apart from one version 1 in which, smartphone “reads the step number” as the user walks. In version 2, we have implemented the second risk-mitigation strategy devised for a risk in our running example, “screen tapping”, where the user taps the screen for every step taken. In version 3, “phone vibrating” for every step taken is implemented. These three versions of Theia are tested with approximately 60 students (20 students for each version).

For the experimentation, undergraduate, masters-level and Ph.D.-level students, aged around 18-40, were involved as subjects, who were blind-folded and asked to use Theia for performing basic activities such as speaking out his/her current location, choosing a destination, walking forward, stopping at a specific point, turning at the right place in the right direction as and when indicated by the application. The experimentation took approximately 1 hour in total for each participant for monitoring the usage of app and gathering the feedback from the participant, concerning the general feelings/comments, the risk-mitigation features implemented, the weaknesses/limitations and recommendations. Two students were present (one ahead and one behind) all the time within a proximity to the participant to ensure the safety of the student while walking.

Fig. 5 explains the results of the experience of the students using each of the risk mitigation strategies for addressing the risk “Incorrect number of steps walked”. When students tested the feature of phone vibration, for each step taken, there were both good and bad observations. The bad observations were that some students were partly able to follow the vibration given by the phone for every step taken. While using this feature, a few students were paying more attention to confirm if the phone has given a vibration or not. In that confusion, they missed route, walked in zig-zag pattern, walked too fast sometimes, stopped walking, and just moved hands, etc. which resulted in some good number of risks. Similarly, some students who focused on the feature well were able to use the most part of feature without committing any mistakes, hence alleviating some risks.

There were both positive and negative observations with the students that tested screen tapping for every step taken to keep track of number of steps. Students were more confident when they used this feature. However, there were some risks that were faced by the user as a result of their actions (e.g., not paying attention to the step number, etc.). Some users did not tap the screen accurately, some tapped the screen twice for one step taken, and some did not tap in the right...
spot on the screen which resulted in the step not being counted or being counted more than once respectively, which resulted in walking incorrect number of steps forward and making incorrect turns. Users who walked very slow to tap the screen for every step taken took more navigation time to reach their destination.

For the “smartphone reading the step number” for every step taken, the students were much more comfortable and confident using this feature compared to the other two. Since the steps taken were read out loud, most of the students had their focus on the steps they were taking. There were a smaller number of errors committed, hence a smaller number of risks. When the background noise was too loud, some students were not able to follow the step number and missed the route, walked in a zig-zag pattern trying to pay attention to the volume. Some students used old phones whose volume was not clearly audible. Those students were more comfortable using the screen tapping feature.

Overall, based on our observation, the students committed a smaller number of risks and alleviated a greater number of risks when the phone read the step numbers for the users. Most of them were able to reach their destination in a timely manner. The only issues with this feature were some hiccups with the hardware and hardware behavior.

**Threats to Validity:** The results from the risk mitigation strategies depends on the individuals (students) who participated in the experiment using three versions of Theia. The experience of using the Theia app with various risk-mitigation strategies also varies from individual to individual. We are yet to receive our IRB approval to test our smartphone application with real blind people. We feel that testing with real blind people may give us an edge over blind-folded people, especially with identifying a variety of risks they face and while using the app overall.

**V. Conclusion**

In this paper, we have proposed a goal-oriented framework, Murphy+G, which extends the NFR Framework with the Reference Model for identifying risks and risk-mitigation strategies for collaborative systems. More specifically, this paper has presented 1) an ontology which comprises of key concepts such as action, risk, agent, etc.; 2) Capture most important NFR softgoals by using a questionnaire, 3) Identify risks associated with the softgoals while devising risk-mitigation strategies, 4) Extend the NFR Framework with the Reference Model and 5) Perform trade-off analysis to identify the best solution to help alleviate those risks. We used three versions of Theia to see the strengths and weakness of the Murphy+G framework, and we have observed that the features implemented as risk mitigation strategies have indeed increased the confidence of the users to reach their destination in a safe, timely and reliable manner. As future work, we plan to apply our approach to a variety of domains which involve collaboration between the user and the software system (e.g., autonomous vehicles domain) for performing risk analysis and providing risk mitigation strategies. A graphical tool for risk analysis is also underway. Experimentation and validation of the risk mitigation strategies and different versions of Theia developed must be tested with real blind people upon the IRB approval as real blind people can give the authors much better insights of what changes must be made to make the app help meet blind people’s goals.

**References**


AFL\textsuperscript{2}-oop: Loop Coverage Guided Greybox Fuzz Testing

Haochen Jin, Liwei Zheng, Zhanqi Cui*  
Computer School, Beijing Information Science and Technology University, Beijing, China  
Email:{jinhaochen0325, zlw, czq}@bistu.edu.cn

Abstract—Fuzz testing automatically generates and executes test cases, to detect more defects by covering more logical and state spaces of the program under test (PUT). However, it becomes more difficult to adequately test the PUT with increasing size and code complexity. Studies have shown that complex code is more likely to contain defects, and the loop is one of the main reasons for increased code complexity. Therefore, it is necessary to thoroughly test the loops, but existing fuzzers cannot focus on the loops of the PUT. To address this issue, we design a loop interval coverage metric to measure the testing adequacy of the loop. Additionally, we propose a greybox fuzz testing approach named AFL\textsuperscript{2}-oop (AFL for Loop), which uses loop coverage as guidance. First, we analyze the loops of the PUT and expand the bitmap. Then, fuzz testing is guided by loop interval coverage and branch coverage. A prototype tool is implemented based on the proposed method, and experiments are carried out on four real-world software programs, such as LibXml2, LibMing, etc. The results show that AFL\textsuperscript{2}-oop achieves higher coverage, triggers more crashes, and reproduces defects faster than AFL and FairFuzz.

Index Terms—fuzz testing, coverage-based greybox fuzzing, loop analysis, crash reproduction, defect detection.

I. INTRODUCTION

Software defects are faults, errors, or failures in software [1], which can result in unexpected outcomes [2] and potentially threaten people’s safety or property in severe cases [3]. Software testing is used to detect potential defects in the program under test (PUT), but manually designing and executing test cases consumes much manpower and is inefficient. Fuzz testing [4] detects software defects by automatically or semi-automatically generating test inputs and monitoring the runtime status of the PUT. Fuzz testing, which reduces manual testing costs and improves testing efficiency, has become an effective method for detecting defects in real-world software [5]. Among them, greybox fuzz testing is not only more scalable but also combines the advantages of whitebox and blackbox fuzz testing [6], and has been widely studied in recent years [7] [8]. Typically, fuzzers with higher coverage can detect more defects in the PUT [6], because the coverage is closely related to the defect detection rate. Therefore, many greybox fuzzers try to increase the coverage of the logical and state space of the PUT as much as possible within a limited time.

As the size and code complexity of software increase, it becomes more difficult to achieve adequate testing coverage [9]. If limited resources are allocated equally to all the code, fuzz testing will be inefficient and unable to detect hidden defects in the PUT. Studies have shown that complex code is more likely to contain defects [10], Loops cause the number of paths to increase exponentially in software, which is one of the main reasons for code complexity to increase. After using AFL (American Fuzzy Lop) to test LibMing\textsuperscript{2} and Libxml2\textsuperscript{3} for 50 hours, we found that 17% and 50% of defects are located in loops. Testing loops is essential but challenging [11]. The number of loops increases with the code size and complexity of software, which presents a significant challenge for testing. Fuzzers may exhaust all testing resources before achieving high coverage [12]. In addition, existing fuzzers treat loops as simple branches, and existing coverage metrics are too general to evaluate the testing coverage of loops. As a result, it is difficult for fuzzers to adequately test loops in the PUT.

To address this issue, we design a loop interval coverage metric that divides the number of loop iterations into intervals to measure the coverage of the loop. Moreover, we propose an approach that uses loop coverage to guide greybox fuzz testing, which is called AFL\textsuperscript{2}-oop (AFL for Loop). First, it analyzes the loop in PUT and extends the bitmap. Then, it uses the loop interval coverage and branch coverage to guide the fuzz testing. Based on AFL\textsuperscript{2}-oop, we have implemented a prototype tool and conducted experiments on four widely used real-world software. The experimental results show that AFL\textsuperscript{2}-oop not only cover 2.9% and 1.8% more loop intervals than AFL and FairFuzz, but also triggers 33 and 67 more crashes than AFL and FairFuzz. In addition, AFL\textsuperscript{2}-oop costs 4578 and 6089 seconds less than AFL and FairFuzz to trigger six defects, respectively.

In summary, this paper makes the following contributions:

- A loop interval coverage metric is designed to evaluate the coverage of the loops in the PUT. Based on the loop interval coverage metric, a loop coverage guided graybox fuzz testing approach is proposed.
- A prototype tool is implemented based on AFL\textsuperscript{2}-oop and experiments are conducted to compare with AFL and

\*Zhanqi Cui is the corresponding author.

DOI reference number: 10.18293/SEKE2023-075

1 AFL. https://lcamtuf.coredump.cx/afl/
2 LibMing. https://github.com/libming/libming
3 Libxml2. https://gitlab.gnome.org/GNOME/libxml2
FairFuzz on four real-world software in terms of code coverage, defect detection and crash reproduction.

II. LOOP COVERAGE GUIDED GREYBOX FUZZ TESTING

AFL^2oop consists of two phases: pre-processing and fuzz testing, as shown in Fig. 1. In the pre-processing phase, the PUT’s basic blocks are analyzed, the bitmap and source code are instrumented. In fuzz testing, test cases are generated and executed, coverage is analyzed through basic blocks and loop coverage analysis. Interesting test cases are filtered and inserted into the test case queue. When the targets for coverage and number of crash triggers are reached, the testing is terminated and a report is generated.

![Fig. 1: Overview of AFL^2oop.](image)

A. Loop interval coverage metric

Current coverage metrics for fuzz testing mainly measure the coverage of edges, branches, and lines of code in the PUT, but the coverage of loops is usually treated as a subset of branch coverage. This oversimplification limits the accuracy and effectiveness of existing coverage metrics in evaluating the quality and adequacy of fuzz testing results.

The branch condition of a loop determines the execution of statements contained in the loop body. If the condition is true, the statements are executed. Otherwise, the program will skip the loop body and continue with the statement next to the loop. However, using traditional coverage metrics that treat loops as branches and only record whether they are executed or not, as is commonly used in existing fuzz testing methods, ignores the variations due to different loop iteration counts. Instead, focusing on each individual iteration of the loop would be too resource-intensive. For example, in Listing 1, when the loop iterates 11 to 19 times, the same part of the loop is executed. Additionally, all the branch statements of the loop are executed if the loop iterates more than 20 times. However, a large number of loop iterations can cause an explosion of the solution space, leading to significant overhead in tracking loop coverage. To address this issue, a threshold should be set as the upper limit for counting iteration times during loop coverage analysis.

For this reason, we propose a new metric for loop interval coverage, called LoopCover_k, which uses k as the loop interval to evaluate loop coverage. In equation (1), \( L_{\text{max}} \) represents the maximum number of loop iterations counted for measuring loop coverage measurement. \( R = \{ R_1, R_2, \ldots, R_n, \ldots, R_m \} \) (\( 1 \leq n \leq m \)) is a set of loops, where \( R_n \) represents the n-th loop in the PUT. The set \( R_n \) is further defined as \( R_n = \{ r_{1,n}, r_{2,n}, \ldots, r_{i,n}, \ldots \} \) (\( 1 \leq i \leq L_{\text{max}} \)), where \( r_{i,n} \) indicates the coverage of the i-th loop interval. If a test case executes \( R_n \) l times, \( (i-1) \cdot k < l \leq i \cdot k \), then the i-th loop interval of \( R_n \) is considered as covered, and \( r_{i,n} \) is set to 1, otherwise it is set to 0.

B. Loop instrumentation and bitmap extension

Instrumentation involves inserting probes into the PUT without affecting its functionality. These probes collect runtime information during test case execution, which are used by the fuzzers to guide test case generation. For instance, AFL uses instrumentation to track edge coverage by inserting probes into basic blocks during compilation and dynamically computing coverage information of connected edges during test runs. This allows AFL to accumulate edge coverage information during testing.

The loop coverage information is used by AFL^2oop to guide its fuzz testing process. However, storing this information directly on the original shared bitmap of AFL can result in conflicts and errors. To address this issue, we implement an expansion of the shared bitmap from 64KB to 128KB, which was named bit_loop_map. Subsequently, the loops of the PUT are analyzed, and the instrumentation method of AFL is extended to collect the execution information generated by the test cases.

The procedure for analyzing and instrumenting loops in the PUT is outlined in Algorithm 1. The input of the algorithm is a set of functions (\( F \)). The algorithm begins by initializing three data structures: loopMap, f_loop and f_branch (lines 1-3). These data are used to record information about the loops, the precursor loops and the precursor branches, respectively. The algorithm then iterates over the basic blocks in \( F \) (lines 4-17). If a basic block \( BB \) under analyzed is located within a loop (line 5) and it has not been recorded in bit_loop_map the probe code is inserted into \( BB \) via Instrumentation(BB) (line 7). Subsequently, the algorithm searches for the loops and branches to which \( BB \) belongs in f_loop and f_branch, otherwise using null instead (lines 8-9). Next, the types of each basic block (loop, branch and normal) need to be identified. The type of each basic block, along with the loop and branch to which it belongs is stored in loopMap, the loop and branch basic blocks are stored in f_loop and

Listing 1: A code fragment of a loop

```plaintext
1: for (int i = 0; i < n; i++) {
2:     if (i == 5) \[\]
3:         \[\]
4:         if (i == 10)\{
5:             \[\]
6:             if (n < 15)\{
7:                 \[\]
8:             \}
9:         \}
10:     \}
11: }"
LoopCover \_k (R) = \frac{\sum_{n=1}^{m}(k \cdot R_n)}{L_{max}} = \frac{\sum_{n=1}^{m}(k \cdot \sum_{i=1}^{\max} r_{i,n})}{L_{max}} \quad (1)

Algorithm 1. Analyze and instrument the PUT for loop coverage
Input: \( F \) = the set of functions in the PUT.
Output: the PUT instrumented and basic block information of the loops is stored in loopMap.
1: loopMap = Hash()
2: f \_loop = Stack()
3: f \_branch = Stack()
4: for BB in F:
5: if isLoopHeader(BB):
6: if !f \_loop \_map.full() and !loopMap.find(BB):
7: f \_loop.push(BB)
8: BB \_loop = f \_loop \_findLoop(BB)
9: BB \_branch = f \_branch \_findBranch(BB)
10: if isLoopHeader(BB) and !loopMap.find(BB):
11: loopMap.add(BB, "loop", BB \_loop, BB \_branch)
12: if BB \_branch.push(BB)
13: else if (isIf(BB) or isSwitch(BB) and BB \_childNum > 1):
14: loopMap.add(BB, "branch", BB \_loop, BB \_branch)
15: else:
16: loopMap.add(BB, "normal", BB \_loop, BB \_branch)
17: if BB \_loop.full() and !loopMap.find(BB):
18: Instrumentation(BB)
19: BB \_bbCover = min(BB \_bbCover, L_{max})
20: interval = floor(BB \_bbCover / k)
21: if testSetCover \_cover(interval, BB):
22: testSetCover.setLoopCover(interval, BB)
23: return true
24: for child in BB:
25: if !child \_bbCover = 0 and !testSetCover \_cover(0, BB):
26: testSetCover.setBranchCover(0, BB)
27: return true
28: return false

Algorithm 2. Test case selection
Input: testSetCover, loopMap, k, and L_{max}.
Output: test case is interesting or not.
1: for BB in loopMap:
2: if BB.type is loop:
3: BB.bbCover = min(BB.bbCover, L_{max})
4: interval = floor(BB.bbCover / k)
5: if testSetCover.cover(interval, BB):
6: testSetCover.setLoopCover(interval, BB)
7: return true
8: if BB.type is branch:
9: for child in BB:
10: if !child.bbCover = 0 and !testSetCover.cover(0, BB):
11: testSetCover.setBranchCover(0, BB)
12: return true
13: return false

C. Loop coverage guided greybox fuzz testing

The process of determining the interesting test cases according to their loop coverage of the PUT is outlined in Algorithm 2. The inputs of the algorithm include the coverage information of the loop interval and the basic blocks of loops (testSetCover), the result of the loop analysis performed during the pre-processing phase (loopMap), the loop interval (k), and the maximum number of loop iterations (L_{max}). First, the algorithm traverses all BBs in loopMap (line 1). Then, if the type of BB is loop, the algorithm analyzes its coverage by using the number of loop iterations (BB.bbCover, from bit_loop_map) with the limitation of L_{max}, and k is also used to calculate the loop interval coverage of a test case (lines 2-7). If the loop interval in the testSetCover has not been covered by another test case, current test case is considered to be interesting, and the algorithm returns true. If the type of BB is branch, identifying whether the branch has been covered by the test case need to examine all the successor basic blocks of the branch (lines 8-12). If the branch has not been covered in the testSetCover, the test case is considered to be interesting, and the algorithm returns true. Finally, if the test case fails to cover any new loop interval or branch, the algorithm returns false (line 13). During fuzz testing, test cases that cover new edges are considered interesting and added to a queue, which is a circular linked list. The selected test case is executed one at a time and interesting test cases are added to the end of the queue. AFL\_oop’s test case queue takes both loop interval and branch coverage into consideration, in addition to performance metrics.

III. EXPERIMENTS AND EVALUATIONS

A. EXPERIMENTAL DESIGN

1) Subjects & Benchmarks:
Subjects. A loop coverage guided greybox fuzzer is implemented based on AFL\_oop. Two existing greybox fuzzers, AFL and FairFuzz, are selected as subjects for comparison. AFL is an open-source greybox fuzzer that uses code coverage feedback to guide fuzz testing, while FairFuzz automatically adjusts input mutation to test rare parts of the PUT, achieving high coverage without sacrificing efficiency.

Benchmarks. The experiment uses four projects as benchmarks: LibXml2, mJS, LibMing, and lrzip, which cover various formats such as XML, JavaScript, Flash, and zip. These projects range in size from 3.8K to 90K lines of code and are commonly used in fuzzing research [13] [14]. The benchmarks and initial seeds are also used in a previous study. The total size of the five projects is 1.25 million LoC, containing 3659 loops and 79384 branches. The benchmark information is presented in TABLE I.

2) Evaluation Setups:
In experiments, we use three coverage, including branch coverage (BCOV), line coverage (LCOV) and LoopCover\_k to evaluate the coverage of the PUT by testing with different fuzzers. Additionally, we use the number of crash triggers and crash reproduction time to assess the ability of the fuzzers in detecting defects. BCOV measures the coverage of branches, while LCOV measures the coverage of lines of code in the

\[ R = \frac{\sum_{n=1}^{m}(k \cdot R_n)}{L_{max}} \]

\[ R = \frac{\sum_{n=1}^{m}(k \cdot \sum_{i=1}^{\max} r_{i,n})}{L_{max}} \]

\[ \text{LoopCover}_k(R) = \frac{\sum_{n=1}^{m}(k \cdot R_n)}{L_{max}} = \frac{\sum_{n=1}^{m}(k \cdot \sum_{i=1}^{\max} r_{i,n})}{L_{max}} \]

\[ \text{LoopCover}_k(R) = \frac{\sum_{n=1}^{m}(k \cdot R_n)}{L_{max}} = \frac{\sum_{n=1}^{m}(k \cdot \sum_{i=1}^{\max} r_{i,n})}{L_{max}} \]
PUT. BCOV and LCOV are also widely used in other fuzz testing studies [3] [15]. The metric LoopCover₅, introduced in this paper is used to evaluate the coverage of the loop in the PUT, and L̄₉₉ is set to 100 in the experiment. To collect loop execution information, an additional function is introduced into AFL and FairFuzz [5] without disrupting their core functionality. The experimental environments are as follows: 64-bit Ubuntu 18.04, with AMD Ryzen7 5800X @ 3.8 GHz CPU, and 64 GB RAM.

3) Research Questions:

To evaluate the performance of AFL²-oop, the following two research questions are addressed in the experiment:

- **RQ1.** How does AFL²-oop compare to other fuzzers in terms of code coverage?
- **RQ2.** How effective is AFL²-oop in detecting defects compared to other fuzzers?

### B. Experimental Results and Discussions

1) **RQ1.** How does AFL²-oop compare to other fuzzers in terms of code coverage:

To evaluate the performance of AFL²-oop, we carry out an experiment to compare it with FairFuzz and AFL.

The results of the evaluation of the performance in terms of BCOV and LCOV are shown in Fig. 2 and Fig. 3, respectively. The results indicate that the BCOV and LCOV of the four PUTs increase over time when using different fuzzers. Among the fuzzers, AFL²-oop achieve the highest BCOV and LCOV when testing Libxml2, LibMing and lrzip, but the BCOV and LCOV of testing mJS by AFL²-oop are slightly lower than FairFuzz. Specifically, AFL²-oop’s BCOV and LCOV are 0.8-4.3% and 0.1-0.8% greater than FairFuzz and AFL, respectively. However, FairFuzz achieves the greatest BCOV and LCOV when testing mJS. The results indicate that AFL²-oop can cover more branches and more lines of code in the PUTs by using loop coverage guided greybox fuzz testing.

In addition to evaluate the performance of AFL²-oop with BCOV and LCOV, we also compare its performance to AFL and FairFuzz with respect to LoopCover₅. The results of this comparison are shown in TABLE II. Overall, AFL²-oop achieves the highest LoopCover₅ among the four PUTs, with an average increase of 2.9% compared to AFL and 1.8% compared to FairFuzz. For example, in the case of Libxml2, AFL²-oop achieves 77.9% LoopCover₅, which is 3.0% and 5.3% greater than FairFuzz and AFL, respectively. This is due to the fact that AFL²-oop uses loop coverage guided greybox fuzz testing. Test cases that cover new loop intervals are selected and inserted into the test case queue.

2) **RQ2.** How effective is AFL²-oop in detecting defects compared to other fuzzers:

TABLE III gives the results of comparing the defects detection ability of AFL, FairFuzz and AFL²-oop. The experiment count the number of unique crashes triggered by the three fuzzers for the four PUTs. The results show that AFL²-oop outperform both AFL and FairFuzz in terms of the number of unique crashes triggered. AFL²-oop triggers 483 unique crashes for the four PUTs, which is 33 more than AFL and 67 more than FairFuzz. AFL²-oop also triggers more unique crashes than AFL and FairFuzz for each individual PUT.

Additionally, we analyze the total number of defects detected by the three fuzzers in the four PUTs as well as the defects located in loops. Our analysis reveal that 29 defects are detected by the four PUTs, with 4 (17%) and 2 (50%) defects of LibMing and Libxml2 are located in loops, respectively. TABLE IV shows the time required for AFL, FairFuzz, and AFL²-oop to reproduce the six defects. AFL²-oop is able to reproduce four out of six defects in the shortest time, with a total time cost of 11557 seconds. AFL and FairFuzz can only reproduce one crash each in the shortest time, with total time costs of 16135 and 17646 seconds, respectively. For example, AFL²-oop reproduces the defect decompile.c:2015:37 of LibMing in 2629 seconds, which is 487 seconds faster than AFL, while FairFuzz is unable to detect and reproduce this crash. AFL²-oop also reproduce the defect valid.c:772:30 of Libxml2 in only 621 seconds, which is 2435 seconds and 564 seconds faster than AFL and FairFuzz, respectively.

Answer for RQ2. The experimental results show that AFL²-oop outperforms AFL and FairFuzz in terms of both the number of crashes triggered in the PUT and the time spent to reproducing crashes caused by defects in the loops. AFL²-oop trigger 483 crashes in the four PUTs, which is 33 more than AFL and 67 more than FairFuzz. In addition, AFL²-oop spent 39.6% and 68.2% less time than AFL and FairFuzz to reproduce the crashes caused by the defects located in the loops.

IV. DISCUSSIONS

A. Effects of loop interval

The results presented in Section III. indicate that AFL²-oop not only improves testing coverage but also detect more defects of the PUTs. The loop interval (k) is a crucial parameter of AFL²-oop. To assess the impact of k on AFL²-oop,
TABLE III: The number of unique crashes triggered by AFL, FairFuzz and AFL2oop.

<table>
<thead>
<tr>
<th>Programs</th>
<th>AFL</th>
<th>FairFuzz</th>
<th>AFL2oop</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibMing</td>
<td>349</td>
<td>319</td>
<td>360(+11)</td>
</tr>
<tr>
<td>Libxml2</td>
<td>100</td>
<td>87</td>
<td>118(+18)</td>
</tr>
<tr>
<td>mJS</td>
<td>1</td>
<td>10</td>
<td>5(-5)</td>
</tr>
<tr>
<td>lrzip</td>
<td>0</td>
<td>0</td>
<td>0(0)</td>
</tr>
</tbody>
</table>

TABLE IV: The time required for AFL, FairFuzz and AFL2oop to reproduce the crashes of defects in the loops (in seconds).

<table>
<thead>
<tr>
<th>Programs</th>
<th>Defects trigger crashes</th>
<th>AFL</th>
<th>FairFuzz</th>
<th>AFL2oop</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibMing</td>
<td>outputscript.c:1440:17</td>
<td>2150</td>
<td>3098</td>
<td>1595(-555)</td>
</tr>
<tr>
<td></td>
<td>decompile.c:2015:37</td>
<td>3116</td>
<td>T/O</td>
<td>2629(-487)</td>
</tr>
<tr>
<td></td>
<td>decompile.c:1843:74</td>
<td>2722</td>
<td>3958</td>
<td>3044(+322)</td>
</tr>
<tr>
<td></td>
<td>decompile.c:1843:64</td>
<td>3522</td>
<td>2385</td>
<td>2733(+348)</td>
</tr>
<tr>
<td>Libxml2</td>
<td>valid.c:772:30</td>
<td>3056</td>
<td>1185</td>
<td>621(-564)</td>
</tr>
<tr>
<td></td>
<td>valid.c:729:30</td>
<td>1569</td>
<td>1762</td>
<td>935(-624)</td>
</tr>
</tbody>
</table>

experiments were conducted with $k$ is set to 50, 20, 10, 5, and 1, respectively. The results show that despite the values of $k$ are varied, both BCOV and LCOV of the four PUTs are increased with increasing of testing time. Among them, Libxml2, LibMing, and lrzip achieved the highest BCOV and LCOV when $k = 5$. The BCOV ($k = 5$) of mJS is slightly lower than $k = 100$, but the LCOV ($k = 5$) still exceeded that of all other $k$ values.

As a result, loop interval for AFL2oop is set as $k = 5$ in the experiments, as it leads to a higher coverage of branches and lines of both code compared to other values of $k$. Specifically, setting $k = 1$ generates numerous invalid test cases, while using excessively large intervals like $k = 50$ or $k = 20$ may miss interesting test cases and decrease coverage.

B. Threats to validity

Internal validity. The threats to internal validity mainly lie in the implementation of the studied fuzzers in our evaluation. To ensure accuracy, AFL was used as the basis for code instrumentation, bitmap extension, and fuzz testing. Moreover, the study used the same initial seed test cases as other studies to mitigate internal validity threats [13] [14]. The experiments were repeated five times and averaged to reduce the impact of randomly mutated test cases.

External validity. The threats to external validity mainly lie in the subjects and benchmarks. To ensure external validity, representative fuzzers AFL and FairFuzz [5] are used in the study, as they have been widely used and studied [13] [16]. Four widely-used programs of varying sizes and formats are used as benchmarks, including Flash, XML, JS, and zip. These benchmarks have been used in previous studies [13] [14] [17], ensuring that the results can be generalized to a wide range of software.

Construct validity. The threat to construct validity mainly lies in the metrics used in the experiments. Metrics such as branch coverage (BCOV), line of code coverage (LCOV), number of crashes triggered, time to reproduce crashes caused by defects located in the loop, and loop iteration interval coverage are used to evaluate the performance of fuzz testing tools. afl-cov\(^6\) is used to obtain branch and line of code coverage, while the number of defect triggers and time to reproduce defects are evaluated using AFL ucrash and AddressSanitizer\(^7\). Loop iteration interval coverage is also proposed in the paper to evaluate the testing coverage of loop structures.

V. RELATED WORK

Greybox fuzz testing, an integration of blackbox and whitebox fuzz testing, has gained significant attention as a practical and effective software testing method [18]. Greybox fuzzers use an evolutionary algorithm to generate new inputs and traverse paths in the program under test, guided by feedback information obtained from its execution. This approach can

---


7 AddressSanitizer. https://github.com/google/sanitizers/wiki/AddressSanitizer
be further categorized into coverage-guided and target-guided methods, depending on the approach used to guide the fuzzing process.

AFL and UnTracer [19] are coverage-based fuzz testing tools that collect execution information during testing to generate test cases based on coverage variants. UnTracer improves the coverage detection algorithm to enhance efficiency. Coverage-guided methods provide comprehensive coverage of the program within a specified test time frame, but may waste test resources on unimportant code.

 AFLGo [14] and FairFuzz [5] are target-oriented fuzz testing tools that aim to cover target code blocks or functions. AFLGo calculates distance to the target block and adjusts the test case generation strategy, while FairFuzz prioritizes the exploration of less-visited parts of the program under test and adjusts the byte-level variation method.

Loop structures in code can lead to increased complexity and latent defects [20], but previous work in fuzz testing has not fully addressed the testing of these structures. Studies take into account the testing of these structures through techniques such as loop unwinding, invariant inference, and summarization [22]. However, machine learning-based defect prediction models still require manual confirmation and can result in a high false alarm rate, while symbolic execution methods can have limited scalability.

The proposed grey-box fuzz testing approach, AFL^2oop, focuses on the loop structure of the program under test as its target. It designs a loop interval coverage metric and derives its fuzz test by taking the coverage of the loop interval of the test case into consideration, ensuring adequate coverage of the loop structure while improving performance testing loop structure by dividing the loop into intervals.

VI. CONCLUSIONS

In this paper, we design the loop interval coverage metric as to measure the testing coverage of the loops in the PUT by fuzzers. Furthermore, we introduce AFL^2oop, which uses loop interval coverage metric to guide greybox fuzz testing. Based on the proposed approach, a prototype tool was implemented and compared with AFL and FairFuzz. The results show that AFL^2oop can cover more lines of code, branches, and loop intervals of the PUT. Furthermore, the number of crashes triggered and the efficiency of reproducing defects are also outperforms the other two fuzzers. In the future, we plan to encapsulate AFL^2oop as a plug-in to be integrated with other commonly used fuzzers to improve the testing efficiency and coverage of loops.

ACKNOWLEDGEMENT

This work was supported in part by the Jiangsu Provincial Frontier Leading Technology Fundamental Research Project (BK20202001), the National Natural Science Foundation of China (No. 61702041), and the Beijing Information Science and Technology University “Qin-Xin Talent” Cultivation Project (No. QXTCP C201906).

REFERENCES


255
Research on Directed Grey-box Fuzzing Technology Based on Target

Liang Sun  
National University of Defense Technology  
Changsha, China  
E-mail:nirvana_sl@126.com

Peng Wu  
National University of Defense Technology  
Changsha, China

Shaoxian Shu  
Hunan Institute of Traffic Engineering  
Hengyang, China

Abstract—In recent years, grey-box fuzzing has been proven to be the most effective method for discovering vulnerabilities in software. However, the present grey-box fuzzing still has some shortcomings. Most existing grey-box fuzzers are coverage guided and consider the program code equally, and spend a lot of time on improving the code coverage. However, most of the code in the program does not contain bugs and only a small percentage of the code may have bugs. Therefore, blindly improving code coverage can waste limited resources on a large number of bug independent locations and reduce the efficiency of fuzzing.

In order to solve the above problems, we propose a targeted mutation strategy for continuous target exploration. By identifying the key bytes in the input seeds, a mutation algorithm for different stages of mutation is proposed to ensure that the subsequent generation of seeds can still hit the target location as much as possible, to realize the continuous exploration of the target location. In addition, based on this mutation strategy, we propose a fuzzing optimization method based on multi-factor seed selection, using LLVM framework to insert the target location information into the test program for preprocessing, and then the multi-factor seed selection strategy is used to select better seeds to explore the target location.

Keywords-fuzzy testing; directed grey-box fuzzing; seed mutation strategy; seed selection strategy

I. INTRODUCTION

Vulnerability mining techniques have received a lot of attention due to the increase in the number of vulnerabilities and the intensification of the damage. Among the many vulnerability mining techniques, fuzzy testing techniques [1] have been proven to be one of the most effective techniques for detecting software security [2, 3], which was first proposed by Miller et al. in 1990. It can be generally divided into white-box fuzzy testing, black-box fuzzy testing, and gray-box fuzzy testing [4] which gray-box fuzzy testing has been proven to be efficient and effective.

There are also some issues in gray-box fuzzy testing. American Fuzzy Loop (AFL) [5], the most representative gray-box fuzzy testing tool in the industry, which does not differentiate the code in the process of fuzzy testing and does not guide the direction of variation, which makes the variation of AFL random and blind.

Based on the widely used AFL tool, this project identifies the high-risk target locations in the program to be tested through manual analysis or static analysis reports, selects the seeds that can easily hit the target locations during the fuzzy testing phase, and "controls" the seed variants so that the subsequent variants can still reach the target locations as much as possible.

II. RELATED WORKS

Directed fuzzy testing is a vulnerability detection technique for target locations in a user-specified program. Unlike coverage-oriented fuzzy testing, directed fuzzy testing spends a lot of time exploring a given target location in the code rather than wasting a lot of time on irrelevant areas of program code. Most existing directed fuzzy testing tools are based on symbolic execution, which uses directed symbolic execution techniques to transform the reachability problem of reaching a target location into an iterative constraint solving the problem for the ultimate purpose of directed fuzzy testing. The effectiveness of directed symbolic execution comes at the cost of efficiency, which spends a significant amount of time on program analysis and constraint solving. In each iteration, directed symbolic execution uses program analysis to determine which paths better approach the target location, constructs corresponding path conditions based on the sequence of instructions along those paths, and uses a constraint solver to check the satisfiability of those conditions.

Directed gray-box fuzzy testing (DGF) is a vulnerability detection technique based on gray-box fuzzy testing to implement directed DGF, which retains the efficiency of gray-box fuzzy testing and usually leaves all program analysis in the program compilation phase. Once the target location is marked, DGF needs to generate seed inputs to reach the target location. In addition to marking the target locations, researchers have also noticed that the interrelationships between targets also help to reach the targets.

V-Fuzz[6] is oriented to vulnerability probabilities, and deep learning models predict vulnerability probabilities to guide the fuzzing process to potentially vulnerable code regions. semFuzz[7] can automatically recover knowledge related to vulnerabilities from text reports and use this information to guide the system in building test cases to trigger known or related unknown vulnerabilities. Seed inputs that perform well in DGF can bring the fuzzy testing process closer to the target location and improve the performance of the subsequent mutation process. Studies have shown that the classical directed fuzzy testing tool AFLGo[8] generates a large number of inputs that are unable to reach the location of the code with vulnerabilities. Therefore, optimizing input generation can be
of great help to improve the directionality of DGF. seededfuzz improves the initial seed generation and selection for directed fuzzing. It uses static analysis, dynamic monitoring, and symbolic execution techniques on the target program to select and generate appropriate seeds for directed fuzzing, it identifies the vulnerability-sensitive parts of the input seeds, generates new inputs by changing the relevant bytes, and feeds them to the target program to trigger exceptions. FuzzGuard[9] uses a deep learning-based approach that filters out inputs that cannot reach the target location before learning, and uses a large number of inputs marked as reachable to train the model. The model is then used to predict the likelihood that the newly generated inputs will reach the target location without running them directly, thus saving time on the actual execution. SemFuzz uses information (system calls and parameters) retrieved from CVE descriptions and git logs to generate seed inputs to increase the probability of hitting the vulnerability function. TIFF and ProFuzzer identify input types to help mutate and maximize the likelihood of triggering memory corruption errors.

Selecting seeds that better hit the target location is a key part of DGF. AFLGo generates the call graph and control flow graph of the program to be tested at compile time to calculate the distance of the seed to the target basic block, prioritizing seeds that are close to the target location. RDFuzz[10] combines distance and frequency to prioritize the seeds. One drawback of the distance-based approach is that it focuses only on the shortest distance, and when there are multiple paths to the same target, seeds that are farther away may be ignored.

III. TARGET-GUIDED DIRECTED GRAY BOX FUZZY TESTING METHODS

Most existing gray-box fuzzy tests are code coverage guided [2], where the most classic coverage oriented tool, AFL, is designed to find more vulnerabilities by increasing the code coverage. In general the broader the scope of code covered during the execution of fuzzy tests, the more likely it is to find potential vulnerabilities in the code. However, most areas of code in real software are not vulnerable, and only some areas may be vulnerable. Therefore, all code should not be treated equally, and more resources should be given to code locations with higher risk or user requirements to explore, instead of wasting a lot of time and computational resources in unrelated code areas, which leads to a decrease in the efficiency of fuzzy testing. Yet there is randomness and blindness in the seed variation part of fuzzy testing that generates a large number of inputs. This leads to the fact that it is often difficult for fuzzy tests to ensure that the mutated inputs can still reach the target location. In particular, when encountering some branches composed of complex conditions, it is more likely to be difficult to reach the target position, and even if the target position is occasionally hit, it is difficult to reach it again. Moreover, most of the vulnerabilities that exist in actual software are not of the type that will be triggered simply after execution, which requires fuzzy testing tools not only to generate seed inputs that can reach the target location, but also to generate a large number of inputs that can continuously reach the target location after seed mutation to achieve continuous exploration of the target location.

A. A Goal-Oriented Directed Gray-Box Fuzzy Testing Framework

This paper improves on the classical fuzzy testing tool AFL by improving the source code preprocessing, seed selection strategy and seed variation method, and proposes the goal-oriented directed gray-box fuzzy testing tool DTFuzz. It can be seen that the entire directed gray box fuzzy testing framework is divided into two phases: the preprocessing phase and the fuzzy testing cycle phase, in which the fuzzy testing cycle mainly includes the multi-factor seed selection strategy and the directed mutation strategy.

After obtaining the source code of the program to be tested, the first thing that needs to be determined is which locations are the target locations in this test. This part can be analyzed by program static analysis tools such as Cppcheck, Clang, etc., and the generated results are processed to generate a uniform format target file, or the target location information can be determined by the user directly specifying the target location.

Next, we move to the fuzzy test loop section, where the inputs are the staked binary program, the initialization seeds, the target location information (actually present in the staked program), and the outputs of the fuzzy test loop phase are the seeds that cause abnormal program behavior (e.g., crashes or timeouts). The priority queue of seeds is obtained through a multi-factor seed selection strategy giving higher variant priority to the preferred seeds. This multi-factor seed selection strategy first distinguishes between seeds that hit the target location and seeds that do not. If the seed does not hit the target, it will use the original AFL strategy, and if the seed can hit the target location, it will have a higher priority. Finally, there is the targeted mutation strategy for the continuous exploration of the target, which can be partly generated not only by mutation DTFuzz first mutates each byte of the seed (using AFL's deterministic mutation phase method) to run the program and determine whether the mutated seeds can hit the target location. For those seeds that can hit the target location, DTFuzz records the location and the mutation method used to mutate these seeds during the mutation process.

B. Targeted Variation Strategies for Targeted Continuous Exploration

The various strategies of both the gray-box fuzzy testing tool AFL and the directed fuzzy testing tool AFLGo do not take into account the use of existing seed variation information, and each variation is performed randomly. AFL's fuzzy testing strategy is to increase the coverage of the code as much as possible, and when the seeds hit the target location, AFL marks them as executed, giving them less energy for subsequent seed prioritization and energy scheduling, and it will explore other branches to discover new paths.

To solve this problem, this paper proposes a target continuous exploration (TCS) for target continuous exploration. Firstly, the key nodes that affect the input seeds to reach the target location are identified by analysis, and it is recorded which variation can still hit the target location, so that in the subsequent variation process, the mutated seeds can be
guaranteed to hit the target location area as much as possible, thus achieving the continuous exploration of the target location and improving the efficiency of the directed fuzzy test.

C. Keyword Section Determination

First of all, it should be clear that the directional variation proposed in this chapter is used only for seeds that can hit the target location. The variation strategy is still used for seeds that fail to hit the target location, while the original AFL variation strategy is still used for seeds that fail to hit the target location.

What prevents fuzzy tests from reaching the target location is usually caused by conditional statements that may be directly related to some bytes of the input or obtained through a series of complex operations. Analyzing the relationship between the input and the conditionals can cause a lot of overhead and thus reduce the performance of the fuzzy test. Efficiency, if the mutated seed can still hit the target location during the execution, by recording the location where this mutation occurs and the corresponding mutation mode, the connection between the input and the conditional branch can be established to some extent, to achieve the purpose of continuous exploration of the target.

D. Directional Variation

After determining the input keyword stanza it goes to the mutation phase of the fuzzy test. For deterministic mutation, not only one byte at a time is mutated, but the step size of each mutation is continuously increased. Obviously, for single-byte mutations, the mutation can be performed directly based on the content of the record, while the situation is slightly different when the mutation step is multiple bytes. Multi-byte mutations can only be mutated if all the bytes they cover can be mutated in the same way, otherwise they are not mutated.

When random mutation is performed, the mutation method used for this mutation is determined first (mutation methods using a combination of mutation operations are considered in order), and the range of bytes that can be mutated can be calculated based on the previously recorded information, so that when the mutation range is randomly selected, the subrange of the mutable byte range can be selected. If such a variation range cannot be found this variation is skipped to proceed to the next cycle of the random variation stage.

By processing the deterministic and random mutation stages as described above, it is possible to generate as many seeds as possible in the mutation stage to reach the target location, especially for those seeds that have already hit the target location, and to mutate the other bytes of the seeds without affecting their hitting the target location again, so that the target location can be fully explored continuously.

### Table I. Continuous Exploration Ability of Targets After Variation Strategy Improvement

<table>
<thead>
<tr>
<th>Actual Software</th>
<th>Target hits</th>
<th>AFLGo</th>
<th>TCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>readelf</td>
<td>44.7/sec</td>
<td>46.4/sec</td>
<td></td>
</tr>
</tbody>
</table>

The results are shown in Table I. For comparison, the rate of hitting the target location was recorded to show the continuous exploration capability of the target location. It can be seen that the exploration ability of TCS to the target location has a certain improve.

IV. Fuzzy Test Optimization Method Based on Multi-Factor Seed Selection

In the previous chapter, the variation algorithm for target continuous exploration (TCS) was introduced, and this method can effectively improve the continuous exploration of target locations. In fact, a lot of work is needed to assist before seed variation can be performed. This chapter optimizes the fuzzy testing procedure of AFL based on TCS, which consists of the following parts: pre-processing of the procedure for the target location and multi-factor seed selection.

A. Multi-Factorial Seed Selection Strategy

AFL’s original seed selection strategy was to select small, fast executing seeds and add them to the “favored” label, based on the code coverage to increase the scope of fuzzy exploration as much as possible to find more potential vulnerabilities, for the targeted fuzzy test can not only consider the code coverage, but also select the seeds that can reach the target location as much as possible to fully explore the target location. Therefore, the selection of seeds that can hit the target, so that they have more possibilities of variation, and thus continue to explore the target location is the basis of the seed selection strategy, and on this basis, further consideration is given to the selection and code coverage of some harder-to-hit targets.

In this section, a multi-factor seed selection strategy (MFS) is proposed to rank the input seed levels. The MFS includes: 1) the number of target functions hit by the seed. 2) the number of hits at the target location. and 3) the path coverage. The execution of fuzzy tests can be easily obtained by taking them in the preprocessing stage using shared memory and thus making the selection. When performing the seed selection, it is necessary to distinguish whether the seeds hit the target location or not. For the seeds that do not hit the target location and the initial input seeds are selected using the original seed selection strategy, and if the seeds hit the target location, MFS is used for seed selection.

B. Number of Seed Hits for the Objective Function

The first factor to consider is the number of target functions hit by the seed. Generally, if the number of hit target locations is higher, the exploration of more target locations can be achieved in a shorter time, thus improving the efficiency of directed fuzzy testing. For complex target locations that are more difficult to hit are ignored if the consideration is whether
the target location is hit or not, so considering the function where the target location is hit can preserve this information and help in the exploration of these targets.

In this section, we need to calculate the number of target functions hit by the seeds, and the distinction between whether the seeds hit the target position function can be done here. We denote this part of the weight as \( W_1 \), the total number of target positions can be obtained directly as \( N \), and the total number of target position functions as \( N_f \), in fact, after distinguishing whether the target function is hit or not, we can use \( N \) to calculate \( W_1 \) directly, if we want to calculate the total number of target functions will bring the extra overhead reduces the efficiency of the fuzzy test. Note that the number of target functions \( N_f \leq N \), the final computed result is somewhat larger but does not affect the prioritization process. Next, the target function of the seed hit the number of hits is labeled as \( n_r \), and the weight of the first factor can be calculated as \( W_1 = n_r/N \). It can be seen that when the total number of targets is small, the number of seed hits on the target function has a large impact on \( W_1 \). Conversely, if there are many targets, a single target hit will have a small impact.

**C. Number of Hits at the Target Location**

Generally speaking, if a target location is hit many times it will be more fully explored, which often means that these target locations are easier to reach. In contrast, locations with fewer hits or even no hits are harder to find if they are potentially vulnerable. This indicates that these targets have a more complex structure or branching conditions. For these targets with fewer hits, they can be referred to as rare targets.

First, we define the concept related to rare targets. The number of executions for a target location \( t \) can be denoted as \( numT[t] \), and for the input seed \( s \) if it hits the target location \( t \), it can be denoted as \( hits(s, t) \). Thus \( numT[t] \) can be computed using equation (1). Thus, by calculating \( numT[t] \), a mapping between the seeds and the number of hits at the target position is established, which is updated each time the seeds hit the target position.

\[
numT[t] = \sum_{s \in S} hits(s, t)
\]  

(1)

After calculating the number of hits per target \( numT[t] \) in this way, the total number of hits for all targets can be further calculated and expressed as \( Num_T \). Based on this we can calculate the rarity of each target position, so that if the seed hits that position, it will receive the corresponding weight. A simple idea is to calculate the proportion of target branches by this, the number of hits of the seed is inversely proportional to its weight, as in equation (3) is shown.

\[
Num_T = \sum_{t \in T} numT[t]
\]

(2)

\[
W_i = \frac{Num_T}{numT[t]}
\]

(3)

However, if we use only this simple method, we may reduce the exploration of rare targets. Under ideal conditions, seeds that can hit rare targets take precedence over other seeds. Therefore, we must design a criterion to distinguish whether a target is a rare target or not. A natural idea is to specify a constant \( n \) and rank the target locations by the number of hits and consider them rare if their rank is lower than \( n \), or when the number of hits is less than a certain percentage of the total number of input seeds.

\[
numT[t] \leq rarity\_cutoff
\]

(4)

\[
rarity\_cutoff = 2^i \text{ such that } 2^{i-1} \leq \min(numT[t])
\]

(5)

For example, if the minimum number of hits on a target location is 31, then any target location with less than \( 2^9 \) hits is considered as a rare branch. Further the rarest target location \( t^* \) can be calculated by equation (6)

\[
t^* = \arg\min(numT[t]) (t \in T)
\]

(6)

Based on the above rare target locations, if a seed can hit a rare target location, it will have a higher priority and will outperform other seeds that do not hit the rare target when prioritizing.

**V. EXPERIMENT AND ANALYSIS RESULT**

Based on the target-guided directed gray-box fuzzy testing approach mentioned in this paper, a directed fuzzy testing tool DTFuzz is designed and implemented based on the classical fuzzy testing tool AFL, which can perform directed fuzzy testing on software with a given source code and target location, and has a significant improvement in the exploration of target location compared with AFL and AFLGo.

**A. Experimental Setup**

We compare DTFuzz with the following fuzzy testing tools: 1) AFL. the classical coverage guided gray-box fuzzy testing tool is also the tool that this tool relies on, which does not care about the target location. 2) AFLGo. is a directed gray-box fuzzy testing tool that uses a simulated annealing strategy to get as close to the target location as possible during the fuzzing process. Since both AFLGo and DTFuzz rely on AFL, we use AFL as the reference benchmark in our experiments. The selection of test cases in our experiments was based on the documentation provided by AFLGo and the Google Fuzzy Test Suite, some of which also provide CVE (Common Vulnerabilities & Exposures) id. from which we selected seven actual open source software to test and evaluate our techniques. The choice of target locations in this experiment was based on the AFLGo documentation and the crash reports from the Google test suite.

**B. Experimental Results and Evaluation**

The main focus in this experiment is on the execution speed of the loop part of the fuzzy test, and the time of the preprocessing part is not taken into account. In addition, code coverage is the classical measure of gray-box fuzzing, and although our method is directed fuzzy testing, code coverage can also reflect the ability of fuzzy testing to a certain extent. Finally, the number of crashes and unique crashes is the ultimate goal of fuzzy testing and an important ability to show the ability of fuzzy testing. Especially for DGF, the main concern is its ability to trigger a crash at the target location.
The relevant statistics are shown in Table III. The statistics here are based on AFLGo, except for the target location hits, which are based on AFL. Where the seed execution speed and code coverage are compared by percentages, and the number of target location hits and unique crashes are multipliers. It can be seen that DTFuzz's execution speed is about 68% of that of AFL except for bmp2tiff and libpng, where it is not too far from AFL. In terms of code coverage, except for objdump, where the coverage is low, all other software have more than 78% of the coverage of AFL. This result shows that by relying on the multi-factor seed selection strategy, it is possible to continue to detect other parts of the program to be tested after the target location is explored more fully. In terms of hitting the target location, all the software has more than 15% improvement except for readelf, which is slightly lower than AFLGo, and some software has 30% improvement above. Finally, the number of unique crashes triggered can be seen to be 1.1 times higher on bmp2tiff than on AFL, and many times higher on all other software, while the number of crashes triggered by DTFuzz can be seen to be the number of collapses is also higher than that of AFLGo, except for transicc.

The most important capability for directed fuzzy testing is the ability to continuously expose a given target location to expose crashes at the target location. We tracked the execution of these unique crashes with DTFuzz, recording the number of crashes at the target location and other locations separately, and the results are shown in Table IV. The experimental results show that most of the unique crashes are triggered at the target location. This means that our technique can be effectively used for targeted fuzzing and more crashes can be detected.

Based on the experimental results, it can be seen that DTFuzz triggers more crashes in xmllint, mjs and objdump than it detects in other software. As an example, xmllint, which parses one or more XML files, is useful for detecting errors in the XML code and in the XML parser itself. It has strict requirements on the input to the program, and if the input does not conform to the format, the program cannot be explored further. This result also shows that the method is able
to break through complex conditional statements and keep exploring the program. It can also be seen that although AFLGo also directs the fuzzy test towards the target location and generates inputs that hit the target location, its performance is not better than DTFuzz because its mutation strategy is the same as AFL, which does not retain the previous mutation information and performs random mutation for the hit target seed.

In addition, further analysis of the effectiveness of the TCS and MFS proposed in this paper is needed. We run DTFuzz with only the multi-factor seed selection strategy (DTFuzz-s) and DTFuzz with only the directed variation strategy (DTFuzz-m) on the software that triggers a crash, where only the software that triggers a crash is considered, and the results are shown in Table V. Based on the experimental results, we can see that DTFuzz-m can trigger more crashes. This is because the mutation strategy ensures that the seeds that can reach the target location are continuously generated in subsequent mutations, generating more seeds that hit the target location and thus triggering more crashes. The number of crashes triggered by DTFuzz-s is similar to that of AFL. This is because for the seed selection strategy, although seeds that are more likely to reach the target location can be selected, giving them more mutation possibilities, there is no guarantee that the seeds generated by mutation can still hit the target location, resulting in fewer crashes triggered. Similarly, without a seed selection strategy, DTFuzz cannot fully utilize the target location information and may waste a lot of resources on irrelevant seeds or repeatedly execute seeds that are more likely to hit the target location, resulting in a decrease in the ability of fuzzy testing.

By comparing DTFuzz-m without the seed selection strategy with DTFuzz, it can be seen that the use of the seed selection strategy on top of the directed variation algorithm improves the coverage of the code by the fuzzy tests and improves the shortcomings of the variation algorithm. This is also in line with our expectation that after the "easier" targets are fully explored, the priority of the seeds that hit these target locations will be lowered to better explore other targets. In particular, the rare targets that are "harder" to hit will be explored continuously, thus improving the ability of the targeted ambiguity test.

VI. CONCLUSION

The goal-oriented directed gray-box fuzzy testing technique based on the randomness and blindness of variation in gray-box fuzzy testing is proposed, which mainly includes the directed variation strategy of goal continuous exploration. It identifies the key nodes linking the input to the target location in the program to be tested by traversing the input seeds, and then uses the directed variation algorithm to handle the deterministic variation phase and the random variation phase to guide the fuzzy test toward the target location, respectively. At the same time, the gray-box fuzzy test is further optimized based on the directed variation strategy by using the LLVM framework to stake the target location information into the program to be tested based on the original staking code, and then using a multi-factor seed selection strategy to select the seeds that are more likely to hit the target. The concept of rare target is proposed to reduce the number of "useless" duplicate seeds generated for the "easier" hit targets, and give higher priority to the harder hit target locations, which is a good way to improve the target location exploration under complex conditions for the directed fuzzy test. The exploration also improves the path coverage.

The method still has the following problems: 1) the target location is still mainly dependent on manual analysis of the program to be measured for the static analysis tool to provide the analysis report due to the number of generally large can not be well used. 2) For how to generate the seeds to reach the target location mainly relies on the random variation process, and no suitable scheme is given to generate the seeds to reach the target location directly.

In Table V, the results of the multi-factor seed selection strategy and the directed variation strategy are respectively displayed. Significant differences were observed in the number of unique crashes. It is predicted that the multi-factor seed selection strategy will greatly improve the code coverage. Based on the experimental results, the degree of code coverage is further explored and improved. The effectiveness of the proposed method is therefore verified. The multi-factor seed selection strategy and the directed variation strategy are effective in improving the code coverage and are therefore recommended for use in future testing scenarios.

REFERENCES

DeepRank: Test Case Prioritization for Deep Neural Networks

Wei Li¹, Zhiyi Zhang¹,²,∗, Yifan Jian³, Chen Liu⁴,∗ and Zhiqiu Huang¹,⁵

¹College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
²Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China
³Science and Technology on Reactor System Design Technology Laboratory, Nuclear Power Institute of China, Chengdu, China
⁴School of Marxism, Yangzhou University, Yangzhou, China
⁵Ministry Key Laboratory for Safety-Critical Software Development and Verification, Nanjing University of Aeronautics and Astronautics, Nanjing, China

wei_sz2116@nuaa.edu.cn, zyzhang10@nuaa.edu.cn, ncepujyf@163.com, lauchan@yzu.edu.cn, zqhuang@nuaa.edu.cn

Abstract—Deep neural networks (DNNs) have been widely used in safety-critical fields such as autonomous driving and medical diagnosis. However, DNNs are easily disturbed to make wrong decisions, which may lead to loss of life or property. Therefore, it is vital to test DNN adequately. In practice, to reveal the incorrect behavior of DNN and improve its robustness, testers usually need massive labeled data to test and optimize DNN. However, labeling test inputs to detect the correctness of DNN predictions is an expensive and time-consuming task that even affects the efficiency of DNN testing.

To relieve the labeling-cost problem, we propose DeepRank, a test case prioritization technique based on cross-entropy loss. The key idea of DeepRank is that the higher the loss value of a test case relative to the DNN, the more likely it is to be mispredicted and the more conducive it is to improve the robustness of the DNN through retraining. Therefore, the cross-entropy loss value can be used for test case prioritization. We experimentally validate our approach on two datasets and three DNNs models. The experimental results demonstrate that DeepRank is significantly better than existing test case prioritization methods regarding fault-revealing capability and retraining effectiveness.

I. INTRODUCTION

Deep neural networks (DNNs) have made breakthroughs in many fields, such as image recognition, speech recognition, and natural language processing. They have been widely integrated into software systems to help solve various tasks, such as autonomous driving systems, medical diagnostic systems, etc. However, DNNs are susceptible to interference to make bad decisions that can lead to loss of life or property, such as the fatal crash of Google’s self-driving car [18]. Therefore, it is vital to ensure the reliability and robustness of software systems driven by DNN.

DNN testing is one of the most effective ways to guarantee its quality [14], [17]. However, unlike traditional software testing, DNN is based on a data-driven programming paradigm that uses massive data to be trained to form internal logic [7], which makes DNN models have the characteristics of poor interpretability and low generalization ability. As a result, many traditional software testing methods cannot be directly applied to DNN testing. To adequately test DNN models, testers often require massive labeled data to test and optimize DNN models. However, labeling test cases to verify the correctness of DNN is costly. There are three main reasons: first, the scale of test cases that need to be labeled is large; Second, it mainly relies on manual labeling, and the labeling efficiency is low; Finally, test case labeling usually requires professional knowledge in specific fields [15].

To relieve the labeling-cost problem, a feasible solution is to prioritize unlabeled test cases and give higher priority to test cases that could lead DNN to make wrong decisions. In this paper, we propose DeepRank, a test case prioritization technique based on cross-entropy loss. The key idea of DeepRank is that the higher the loss value of a test case relative to the DNN, the more likely it is to be mispredicted and the more conducive it is to improve the robustness of the DNN through retraining. Therefore, the cross-entropy loss value can be used for test case prioritization. We only label test cases with high priority after prioritization, which can save labeling costs and improve the efficiency of DNN testing. We designed four sets of experiments for empirical research on two commonly used datasets in image recognition and three DNNs with different structures. The experimental results demonstrate that DeepRank is significantly better than existing test case prioritization methods regarding fault-revealing capability and retraining effectiveness.

The main contributions of this paper are as follows:

• We propose DeepRank, a test case prioritization technique, which can reduce labeling costs and improve the efficiency of DNN testing.
• We use cross-entropy loss value for test case prioritization and prove the effectiveness of our method through experiments.
• The test cases with high priority after prioritization by our method can be used to guide DNN retraining and improve DNN robustness.
The rest of the paper is organized as follows. Section II introduces background on DNN and test case prioritization. Section III describes the implementation of our method. Section IV details the setup of the experiment. Section V analyzes the experimental results and demonstrates the effectiveness of our method. Section VI concludes this paper.

II. BACKGROUND

This section includes basic knowledge about deep neural networks (DNNs), neuron coverage metrics, and test case prioritization methods for DNNs.

A. Convolutional Neural Network

Convolutional neural networks (CNNs) are the core of image processing tasks. As shown in Fig. 1, a CNN consists of multiple layers, i.e., convolution layer, pooling layer, and fully connected layer. The convolution layer is used to extract the features of the input data. The pooling layer is periodically inserted between successive convolutional layers to reduce the number of parameters in CNN and effectively prevent overfitting. The fully connected layer is used to map the learned feature representation to the label space of the input.

Generally speaking, CNN maps the input data \( x \) to the output result \( y \). For example, in an \( N \) classification task, given an input, after processing by CNN internal neurons, an \( N \)-dimensional vector \( \text{out} = \{v_1, v_2, ..., v_N\} \) will be obtained in the output layer and then normalized using the softmax function [2], a set of probability vectors \( \hat{p}_i = \{\hat{p}_{i,1}, \hat{p}_{i,2}, ..., \hat{p}_{i,N}\} \) will be obtained, \( \hat{p}_{i,j} \) represents the probability that the neural network predicts the test case \( x_i \) as an \( j \)-th class, and the final prediction result of CNN is the category corresponding to the value with the highest probability in \( \hat{p}_i \).

B. Neuron Coverage Metric

Inspired by code coverage in traditional software testing, researchers have recently combined coverage with neurons to propose a series of neuron coverage metrics for DNN testing. This section briefly describes the DNN test method guided by the neuron coverage under investigation.

Neuron Activation Coverage (NAC(\( k \))) [12]. The metric defines neuron coverage in DNN as: if the output value of a neuron is greater than the threshold \( k \), the neuron is considered to be covered. The fundamental hypothesis of NAC(\( k \)) is that the greater the number of neurons covered, the more DNN states are explored. For a test case, NAC(\( k \)) is calculated as the ratio of the number of neurons covered by that test case to the total number of neurons in the DNN.

Neuron Boundary Coverage (NBC(\( k \))) [9]. SBC(\( k \)) first counts the upper boundary \( \text{high}_n \) and lower boundary \( \text{low}_n \) of the output value of each neuron in the DNN on the training set. They refer to \((-\infty, \text{low}_n) \cup (\text{high}_n, +\infty)\) as the corner-case regions of a neuron \( n \). This method focuses on measuring test cases coverage in corner-case regions. Since each neuron has one upper bound and one low bound, for a test case, SBC(\( k \)) is calculated as a ratio of the number of neurons covered by a corner-case region to twice the total number of neurons in the DNN.

Strong Neuron Activation Coverage (SNAC(\( k \))) [9]. It can be seen as a special case of NBC(\( k \)) as it only considers coverage of the upper boundary region \( (\text{high}_n, +\infty) \). For a test case, SNA(\( k \)) is calculated as the ratio of the number of neurons covered by the upper boundary to the total number of neurons in the DNN.

Top-\( k \) Neuron Coverage (TKNC(\( k \))) [9]. TKNC(\( k \)) focuses on the \( k \) neurons that are the most active in each layer of the DNN. It is defined as the ratio of the total number of top-\( k \) neurons on each layer to the total number of neurons in the DNN.

C. Test Case Prioritization

Test case prioritization refers to rearranging the execution order of test cases in a test set according to predetermined criteria so that high-priority test cases are executed earlier in the test execution process than low-priority test cases. Two main coverage-based test case prioritization techniques are known as the Coverage-Total Method (CTM) and the Coverage-Additional Method (CAM) [16].

Coverage-Total Method (CTM). The coverage of each test case is calculated first, and then the individual test cases are prioritized based on their total coverage. When multiple test cases have the same coverage, the relative order of these test cases is randomly determined. Assuming that there are \( n \) test cases in the test set \( T \) and \( m \) coverage entities in program \( P \), the time cost of CTM is \( O(nm) \).

Coverage-Additional Method (CAM). The idea of CAM is that if a test case can cover as many entities as possible that were not covered by previously executed test cases, the higher the priority of that test case. Because such a test case is most likely to expose errors not exposed by the previously executed test case. Assuming that there are \( n \) test cases in the test set \( T \) and \( m \) coverage entities in program \( P \), the time cost of CAM is \( O(ma^2) \).

III. OUR APPROACH

We propose a test case prioritization method DeepRank, based on the cross-entropy loss value of test cases. First, we introduce the motivation of DeepRank. Then, we introduce the overall framework of DeepRank. After that, we introduce the specific implementation steps of DeepRank. Finally, we introduce how to use DeepRank to guide the retraining of DNN models to improve their robustness.
A. Motivation

Unlike the DNN test case prioritization method based on uncertainty [3] and neuron coverage [6], [9], [12], DeepRank takes the cross-entropy loss value of a test case relative to the DNN as the prioritize metric, and the greater the loss, the greater the probability of DNN misprediction.

For a trained DNN, the higher the loss value of a test case relative to the DNN, the more likely it is to be mispredicted. For example, as shown in Fig.1, we select two test cases $x_1$ and $x_2$ with actual labels 2 from the MNIST dataset, input them into the trained LeNet1 model and get the prediction results $\hat{p}_1 = \{0.02, 0.03, 0.90, ..., 0.01\}$, $\hat{p}_2 = \{0.01, 0.73, 0.20, ..., 0.02\}$, respectively. Therefore, LeNet1 predicts $x_1$ as 2 but $x_2$ as 1. It can be seen from the prediction results that the $x_1$ prediction is correct, but the $x_2$ prediction is wrong.

$$CEloss = -\sum_{n=1}^{N}p_i,n \log \hat{p}_i,n$$ (1)

According to the cross entropy loss function (CEloss) in (1), where $N$ is the number of output classes. It can be calculated that the loss values of $x_1$ and $x_2$ relative to the LeNet1 model are $loss_1=0.046$ and $loss_2=0.699$, respectively.

From this example, it can be found that the cross-entropy loss value of $x_2$ relative to the DNN is greater than $x_1$, and the probability that the DNN incorrectly predicts $x_2$ is greater than $x_1$. The greater the loss value of a test case for the DNN, the greater the probability of the model misprediction. Therefore, the cross-entropy loss value can be used for test case prioritization. The greater the loss, the higher the priority.

B. Overview of DeepRank

Fig. 2 shows the overall framework of DeepRank. In general, the implementation of DeepRank can be divided into three parts. First, train a model for quantifying the cross-entropy loss value of test cases. Second, input the test cases into the DNN, divided into $N$ sets according to the prediction class of DNN, and the features of the test cases in different sets are extracted in turn. Then the extracted features are input into the cross-entropy loss quantification model constructed in the first step. The cross-entropy loss value of the test case relative to the DNN can be obtained. Finally, prioritization according to the cross-entropy loss value of the test cases. Test cases with large cross-entropy loss values are given higher priority. Then selects high-priority test cases from each category collection to label.

C. Test Case Prioritization Process

- **Step1. Build the dataset:** We use the training set in the original dataset as the initial data because the cross-entropy loss value of each sample in the training set relative to the DNN can be easily obtained, and the training set and the test set are independent of each other. To increase the generalization ability of the cross-entropy loss quantification model, we add the adversarial samples generated by FGSM [4] and PGD [10] into the training set to form a new dataset $X = \{x_1, x_2, x_3, ..., x_m\}$.
- **Step2. Feature extraction:** Let $L = \{e_1, e_2, e_3, ..., e_n\}$ represent the set of neurons in the last hidden layer in the DNN, $\alpha (x)$ represents the output value of the neuron relative to $x$, and $\alpha_L (x)$ represents the Activation Trace (AT) [6] of neurons in $L$, that is, the set of output values of all neurons in the $L$ related to $x$. $\alpha_L (X) = \{\alpha_L (x) \in X\}$ denote the AT of neurons in $L$ on $X$. Then count the range of output values of each neuron $e_i$ in $L$ on $X$: $[low_i, high_i]$, and divide it into $k$ equal intervals $\alpha_E (X) = \{u_1, u_2, u_3, ..., u_k\}$, in this paper, $k = 100$, if $\alpha_{e_i} (x) \in u_j$, let $f_x (e_i) = j$, therefore, for each $x$ in $X$ can extract an $n$-dimensional feature vector $F (x) = \{f_x (e_1), f_x (e_2), ..., f_x (e_n)\}$, where $f_x (e_i) \in [1, k]$.
- **Step3. Extract the label:** After input $X$ to DNN, the output value of neurons in the output layer of DNN is processed by the softmax activation function to obtain the set of predicted probabilities $P (X) = \{p (x) \in X\}$, where $p (x) = \{p_1, p_2, ..., p_N\}$, $N$ represents the number of categories. Then the cross-entropy loss value of $x$ is
calculated according to the actual label as the label of the cross-entropy loss quantification model.

- **Step4. Train model**: Use the features extracted in step 2 and the labels calculated in step 3 as the training set of the model to quantitatively measure the test case cross-entropy loss value. Specifically, DeepRank uses XGBoost [11] to build a cross-entropy loss quantification model, which is one of the most popular algorithms in the field of machine learning with massively parallel computing power and sound portability and can effectively learn more complex features from basic features, so XGBoost is very suitable for solving our problem.

- **Step5. Process unlabeled data**: Input the unlabeled test set \( T \) into the DNN, divide \( T \) into \( N \) sets \( C = \{C_1, C_2, ..., C_N\} \) according to the classification results of the DNN, and then use the method in step 2 to extract the features of \( C_i \) to obtain \( F_C = \{F_{C_1}, F_{C_2}, ..., F_{C_N}\} \), and then input the features into the model trained in step3 to obtain the set of cross-entropy loss values \( \text{Loss} = \{L_1, L_2, ..., L_N\} \) of each test case relative to the DNN.

- **Step6. Sorting and labeling**: Use the quicksort algorithm to sort \( L_i \) in descending order to obtain the sorted unlabeled test set \( R = \{R_1, R_2, ..., R_N\} \). Finally, To evenly select data from each category, according to the test budget, select the first \( n \) test cases from \( R_i \) (\( i=1, 2, ..., N \)) to label.

There are two reasons why features are extracted from AT in L. First, the output of neurons in L is generally regarded as a learned representation of the training data. When the operating context changes, the representation is more stable than the prediction, and this is supported by transfer learning practices, where only the SoftMax layer is retrained for different tasks [5]. Second, DNN prediction comes directly from the linear combination of the output of this layer, so it must be highly correlated with the prediction accuracy [8].

### D. Enhancing DNN with DeepRank

Since DNN is a data-driven programming paradigm, we cannot fix software bugs by directly modifying the code like traditional software development. Still, we can add as much data as possible to the DNN training set and retrain the DNN to enhance its robustness. However, in the actual scenario, a large amount of data collected is unlabeled and requires expensive labeling for DNN retraining. The main idea of DeepRank is that the greater the loss value of a test case relative to the DNN, the greater the probability of DNN misprediction and the more conducive it is to improve the robustness of the DNN through retraining. Therefore, the cross-entropy loss value can be used for test case prioritization. We only label test cases with high priority after prioritization, which reveals more DNN defects within a limited test budget.

In short, DeepRank can not only for test case prioritization but also use test cases with high priority after prioritization by DeepRank add into the training set to retrain DNN to improve the robustness of DNN.

### IV. EXPERIMENTS

This section describes the experimental setup, including the datasets and DNN models used in the experiment, the construction candidate dataset, and the research questions.

#### A. Datasets and Models

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DNN Model</th>
<th>Layers</th>
<th>Neurons</th>
<th>Train set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>LeNet-1</td>
<td>5</td>
<td>42</td>
<td>60000</td>
<td>10000</td>
</tr>
<tr>
<td></td>
<td>ResNet-20</td>
<td>20</td>
<td>698</td>
<td>60000</td>
<td>10000</td>
</tr>
<tr>
<td>SVHN</td>
<td>VGG-16</td>
<td>21</td>
<td>7274</td>
<td>73257</td>
<td>26032</td>
</tr>
<tr>
<td></td>
<td>ResNet-20</td>
<td>20</td>
<td>698</td>
<td>73257</td>
<td>26032</td>
</tr>
</tbody>
</table>

As shown in Table I, to evaluate our proposed method, we selected two widely used public datasets in the field of image recognition: MNIST and SVHN [11], and three DNN models with different structures and scales: LeNet-1, ResNet-20, and VGG-16 [13]. Among them, MNIST is a handwritten digit recognition dataset containing 7,000 grayscale images with a size of 28*28, of which 60,000 are training sets, and 10,000 are test sets, with a total of 10 categories. SVHN was collected from house numbers in Google Street View imagery and contained over 60,000 color images with a size of 32*32. To increase the reliability of experimental results, we selected two different DNN models for each dataset and designed four sets of experiments.

#### B. Construction Candidate Dataset

Although DNNs are carefully trained to predict high accuracy on the original test set, they are highly inaccurate for some corner test cases, so exploring the prioritization of these data is necessary. We use two commonly used adversarial sample generation methods, FGSM [4] and PGD [10], to generate these corner test cases for each dataset. We generate data the same size as the original test set for each adversarial sample generation method and evenly partition the original test set and the generated adversarial data into a new testing set \( T \), and a new validation set \( V \). For the MNIST dataset, we have new test and validation sets of size 15,000 each, where 5,000 are original test sets and the other 10,000 are adversarial samples generated by FGSM and PGD.

#### C. Research Questions

**RQ1. Quantify**: Can DeepRank accurately quantify the loss value of a test case relative to the DNN?

We use the \( R^2 \) Score and the Root-Mean-Square Error (RMSE) to answer RQ1. The \( R^2 \) Score is used to evaluate the regression model’s fitting effect. The higher the coefficient of the \( R^2 \) Score, the closer it is to 1, and the better the fitting effect of the model. The RMSE is used to assess the accuracy of the regression model predictions.

**RQ2. Effectiveness**: Can DeepRank find a better permutation of tests than the baseline methods?
We collect the cumulative sum of the errors found by specific test case prioritization methods and calculate the corresponding RAUC (ratio of area under the curve) between the prioritization method and the theoretical curve.

**RQ3. Enhancement:** Can DeepRank guide the retraining of a DNN to improve its accuracy?

We evenly divide the original test set and the generated adversarial test cases into a test set \( T \), and a validation set \( V \), then prioritize \( T \), and take the top 10\% from the sorted \( T \) to the initial training set for retraining. Finally, we observe the accuracy of the DNN after retraining on the validation set \( V \).

**V. RESULT ANALYSIS**

In this section, we present and analyze the results of our method on the quantification of test case loss values (RQ1) and the effectiveness of other baseline methods in test case prioritization (RQ2), and the improvement of accuracy after DNN retraining (RQ3).

### TABLE II: \( R^2 \) Score and RMSE

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DNN Model</th>
<th>( R^2 ) Score</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>LeNet-1</td>
<td>94.39%</td>
<td>2.33</td>
</tr>
<tr>
<td></td>
<td>ResNet-20</td>
<td>97.9%</td>
<td>2.86</td>
</tr>
<tr>
<td>SVHN</td>
<td>VGG-16</td>
<td>97.07%</td>
<td>3.49</td>
</tr>
<tr>
<td></td>
<td>ResNet-20</td>
<td>98.68%</td>
<td>1.39</td>
</tr>
</tbody>
</table>

**A. RQ1. Quantify**

As shown in Table II, for each dataset and model combination, the \( R^2 \) Score of the test case cross-entropy quantization model trained by us is above 94\%, especially for the SVHN and ResNet20 combination, the \( R^2 \) Score is 98.68\%, which indicates that the model trained with the features we extracted has an excellent fit to the cross-entropy loss of the test case. In addition, we also provide the RMSE for each cross-entropy quantization model, and the results show that the prediction error of the quantization model remains within a low range.

To more intuitively show the effect of the test case cross-entropy loss quantification model, we randomly select 200 test cases from the test set \( T \) for each dataset and model combination. In Fig. 3, the \( x \)-axis represents the number of test cases and \( y \)-axis represents the cross-entropy loss value of the test case. The blue line represents the actual cross-entropy loss, and the orange line represents the cross-entropy loss values quantified by our method. The results show that the cross-entropy loss value quantified by our method is almost consistent with the actual value. It is worth noting that although the cross-entropy loss quantification of some test cases is not accurate enough, as long as it is the same as the actual cross-entropy loss value change trend. Because in practical applications, we only select a small number of test cases with large cross-entropy loss values from massive candidate sets for labeling to save labeling costs.

In summary, DeepRank can accurately quantify the cross-entropy loss value of test cases relative to DNNs within a small error range.

**B. RQ2. Effectiveness**

We compare fault detection rates between DeepRank and other test case prioritization methods. For each dataset and DNN model combination, we calculate the RAUC on the test set \( T \) for each prioritization method. The closer the RAUC is to 1, the better the corresponding prioritization method works. As shown in Table III, DeepRank has achieved excellent results on all datasets and model combinations, i.e., RAUC is above 99\%. In most cases, DeepRank is better than coverage-based and uncertainty-based methods. Taking the MNIST and LeNet1 combination as an example, DeepRank’s RAUC is 0.998, and the NAC(0.75) is only 0.583. The uncertainty-based methods, such as MaxP and DeepGini, were close in all experiments. Overall, the uncertainty-based methods is superior to the coverage-based approaches but worse than DeepRank. Although NBC(0) and SNAC(0) were similar to DeepRank in the MNIST and ResNet20 combination and SVHN and ResNet20 combination, they were far less effective than DeepRank in the other two sets of experiments, indicating that the prioritization effect of NBC(0) and SNAC(0) was unstable. Conversely, the prioritization effect of DeepRank was stable in all experiments.

In summary, DeepRank achieves excellent prioritization effect in all combinations of datasets and models, and in most cases, DeepRank is better than coverage-based and

**TABLE III: THE RAUC OF FAULT DETECTION**

<table>
<thead>
<tr>
<th>Metrics</th>
<th>MNIST</th>
<th>MNIST</th>
<th>SVHN</th>
<th>SVHN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LeNet1</td>
<td>ResNet20</td>
<td>VGG16</td>
<td>ResNet20</td>
</tr>
<tr>
<td>Neuron Coverage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAC(0.75)</td>
<td>0.583</td>
<td>0.854</td>
<td>0.603</td>
<td>0.994</td>
</tr>
<tr>
<td>NBC(0)</td>
<td>0.702</td>
<td>0.975</td>
<td>0.769</td>
<td>0.995</td>
</tr>
<tr>
<td>SNAC(0)</td>
<td>0.605</td>
<td><strong>0.999</strong></td>
<td>0.603</td>
<td>0.995</td>
</tr>
<tr>
<td>TKNC(1)</td>
<td>0.75</td>
<td>0.747</td>
<td>0.753</td>
<td>0.75</td>
</tr>
<tr>
<td>Uncertainty</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxP</td>
<td>0.809</td>
<td>0.845</td>
<td>0.763</td>
<td>0.755</td>
</tr>
<tr>
<td>DeepGini</td>
<td>0.809</td>
<td>0.845</td>
<td>0.763</td>
<td>0.755</td>
</tr>
<tr>
<td>Our</td>
<td><strong>0.998</strong></td>
<td>0.993</td>
<td><strong>0.995</strong></td>
<td><strong>0.996</strong></td>
</tr>
</tbody>
</table>
uncertainty-based methods.

<table>
<thead>
<tr>
<th>Neuron Coverage</th>
<th>Metrics</th>
<th>MNIST LeNet1</th>
<th>MNIST ResNet20</th>
<th>SVHN VGG16</th>
<th>SVHN ResNet20</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAC(0.75)</td>
<td>11.81</td>
<td>57.31</td>
<td>6.69</td>
<td>30.19</td>
<td></td>
</tr>
<tr>
<td>NBC(0)</td>
<td>51.71</td>
<td>55.25</td>
<td>36.89</td>
<td>30.79</td>
<td></td>
</tr>
<tr>
<td>SNAC(0)</td>
<td>25.83</td>
<td>55.39</td>
<td>12.39</td>
<td>30.72</td>
<td></td>
</tr>
<tr>
<td>TKNC(1)</td>
<td>51.3</td>
<td>60.86</td>
<td>32.837</td>
<td>30.8</td>
<td></td>
</tr>
<tr>
<td>Uncertainty</td>
<td>MaxP</td>
<td>38.17</td>
<td>60.61</td>
<td>20.14</td>
<td>12.04</td>
</tr>
<tr>
<td></td>
<td>DeepGini</td>
<td>39.3</td>
<td>60.68</td>
<td>14.13</td>
<td>11.78</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>DeepRank</td>
<td>51.9</td>
<td>61.22</td>
<td>34.41</td>
</tr>
</tbody>
</table>

C. RQ3. Enhancement

For each dataset and model combination, we select the top 10% of the test cases prioritization by each test case prioritization method, add them to the initial training set for retraining the DNN, and then evaluate the effect of retraining by observing the improvement of the accuracy of the model on the validation set \( V \). The results are shown in Table IV, and the accuracy of DNN models can be significantly improved by using DeepRank to guide retraining. For the MNIST and ResNet20 combination, DeepRank can improve the accuracy of DNNs on validation sets by 61.22%. In most cases, DeepRank is more effective at improving the model’s accuracy than coverage-based and uncertainty-based methods. For example, combined with MNIST and LeNet1, DeepRank can improve model accuracy by 51.9%, but NAC(0) improves by 11.81% and MaxP by 38.17%. To some extent, this shows that the greater the loss value of the test case relative to the DNN, the more conducive it is to guide the retraining of the model.

In summary, DeepRank can effectively guide model retraining and improve model accuracy.

VI. CONCLUSION

In this paper, we propose Deeprank, a test case prioritization method based on cross-entropy loss. The key idea of DeepRank is that the higher the loss value of a test case relative to the DNN, the more likely it is to be mispredicted and the more conducive it is to improve the robustness of the DNN through retraining. Therefore, the cross-entropy loss value can be used for test case prioritization. We only label the test cases with higher priority, which can alleviate the cost of test case labeling and improve the efficiency of DNN testing. The experimental results show that DeepRank can effectively quantify the cross-entropy loss of test cases relative to DNN, has an excellent prioritization effect, and can guide DNN retraining, significantly improving the robustness of DNN.

ACKNOWLEDGMENT

This research is supported, in part, by National Natural Science Foundation of China (Grant No.62022162), and Natural Science Foundation of Jiangsu Province, China (Grant No.BK20200442), and Humanities and Social Sciences Fund of Yangzhou University (Grant No. xj2019-07), and Double-Innovation Doctor Program of Jiangsu Province, China (Grant No. (2019) 30755), and “Green Yang Jinfeng Project” Excellent Doctoral Program of Yangzhou, China (Grant No. (2019) 32).

REFERENCES

Code Clone Detection via Software Visualization Representation Learning

Shaojian Qiu, Shaosheng Wang, Yujun Liang  
College of Mathematics and Informatics  
South China Agricultural University  
Guangzhou, China  
qiushaojian@scau.edu.cn

Wenchao Jiang, Fanlong Zhang*  
School of Computer Science and Technology  
Guangdong University of Technology  
Guangzhou, China  
zhangfanlong@gdut.edu.cn

Abstract—Code clone detection technology aims to automatically detect code similarity and help developers identify and reduce code duplication. While code syntax analysis-based methods are commonly used for clone detection, they may not capture semantic information due to bypassing the analysis of code text. To address this issue, this paper proposes a new method called visualization representation learning for code clone detection (VRL4CCD). This method converts source code fragments into grayscale images to preserve textual information and then utilizes VGG16 and a self-attention mechanism to extract features related to code semantic similarity. A siamese neural network is used to learn the similarity pattern between code features. Experimental results on the Big Clone Bench and Google Code Jam datasets demonstrate that VRL4CCD outperforms current clone detection methods regarding precision, recall, and F1-score, indicating the effectiveness of code visualization technology in clone detection tasks.

Index Terms—Code clone detection, software visualization, siamese neural network, attention mechanism

I. INTRODUCTION

During software development, programmers frequently reuse existing code fragments by copying and pasting them, known as code cloning. However, excessive code cloning can result in code redundancy, which increases the maintenance cost of the software system. Moreover, code defects in the software system can spread through code clones. In response to the above problems, researchers try to detect code clone pairs during software development and encapsulate these clone codes to simplify subsequent software maintenance.

Recently, there has been considerable interest in code clone detection methods that exploit deep learning techniques to extract structural-semantic features from code. These methods typically convert code into intermediate forms that contain both syntax and semantic information, such as Abstract Syntax Trees (ASTs), Control Flow Graphs (CFGs), and Program Dependency Graphs (PDGs) [1], [2]. However, these intermediate representations are indirect, and the effectiveness of feature extraction is heavily reliant on the completeness of the information contained in the syntax tree and semantic graph. Moreover, after conversion to these intermediate representations, the textual information in the source code is no longer utilized, which can result in the loss of valuable information needed to evaluate semantic similarity. Additionally, these methods rely on isomorphism techniques to match code subgraphs and global graphs, which can be computationally expensive and time-consuming.

To address the aforementioned issues, we aim to leverage software code visualization technology to enhance the accuracy of code clone detection. Figure 1 serves as a motivating example. It displays two actual code fragments from the Big Clone Bench (BCB) dataset. We wondered if visualizing the code would aid in illustrating the differences between the two programs. To accomplish this, we converted the ASCII decimal value of each character in the source code into rows and columns and interpreted it as a rectangular image. By visually comparing the code images, significant differences between the two programs can be observed. As a result, we conducted experiments to determine whether these characteristics contribute to clone detection.

In this paper, we propose a method called Visualization Representation Learning for Code Clone Detection (VRL4CCD). VRL4CCD first utilizes software visualization technology to convert each source code fragment into a grayscale image at the pixel level, preventing the loss of code information. Secondly, we uniformly reshape the generated code images to a standard size. For images that are smaller than the standard size, we pad them with 0 values. For images that are larger than the standard size, we reshape the code image directly. We then feed the code images of the cloned pairs into a siamese neural network fused with Visual Geometry Group 16 (VGG16) and attention modules to extract semantic similarity features. Finally, we perform a clone detection task using the features generated by VRL4CCD.

Fig. 1. Motivation Case.
The main contributions of our work are as follows:

- We explore the feasibility of code clone detection based on software visualization technology.
- We propose the VRL4CCD method, a code feature extraction method based on code images and siamese neural networks with attention mechanisms.
- We conduct experiments based on two open code clone detection datasets, BCB and Google Code Jam (GCJ). The experimental results show that the precision, recall, and F1-score of VRL4CCD are superior to many current code clone detection methods.

II. RELATED WORK

A. Code Clone Detection

According to the degree of similarity, code clone pairs can be broadly classified into four types [3]. Type-1 (T1) refers to code fragments with the same syntax except for comments and white spaces [4]. Type-2 (T2) corresponds to code fragments with the same grammatical structure but different identifiers, constants, and types [5]. Type-3 (T3) represents code fragments that have undergone modifications after copying, such as changing, adding, or deleting a few statements [6]. Type-4 (T4) denotes code fragments that perform the same function but are implemented using different syntactic constructs, such as bubble sort and quick sort [7]. Since the distinction between T3 and T4 is often ambiguous, researchers have further classified them into three types: strongly type-3 (ST3), moderately type-3 (MT3), and weakly type-3/type-4 (WT3/T4) [8].

With the rise of machine learning and deep learning, many researchers have turned to these techniques for code clone detection. Fang et al. [9] proposed a joint code representation that combines fusion embedding to learn the hidden syntactic and semantic features of source code. Tai et al. [10] developed a tool called CDLH, which utilizes binary Tree-LSTM [11] to encode ASTs and hash functions to optimize the distance between AST vector pairs using hamming distance. Wang et al. [12] extended original ASTs by adding direct control and data flow edges and built a graph representation of programs called Flow-Augmented Abstract Syntax Tree (FA-AST).

B. Software Visualization

Visualization technology has a wide range of applications in software engineering. Tian et al. [13] studied the relationship between software architecture and source code, and they found that nearly 30% of practitioners used software architecture visualization and modeling tools. They believe using these tools can help improve system quality attributes, maintainability, and reliability. Lima et al. [14] designed a software visualization method that graphically shows how code comments are distributed and organized in a software system and interact with the user. Chen et al. proposed software visualization and deep transfer learning for effective software defect prediction (DTL-DP) [15]. DTL-DP visualizes programs as images, applies the self-attention mechanism to extract image features, and feeds the image files into a pretrained, deep-learning model for defect prediction.

Currently, some researchers applied visualization technology in code cloning. Linsbauer et al. [16] propose a visual software reuse method that automatically extracts and combines code to achieve cloning and reuse. Kuar et al. [17] studied the application of machine learning in code clone detection and visual management. Keller et al. [18] proposed the visualization method for code clone detection, where visualization representations of source code are fed into pre-trained image classification neural networks from the field of computer vision. Inspired by these works, we aim to explore further the potential of applying code visualization technology to code clone detection.

III. METHOD

A. Overall Framework

The VRL4CCD method consists of three main steps: (i) generating code images, (ii) constructing a siamese neural network for code feature extraction, and (iii) detecting code clones. Specifically, we start by converting all code fragments into grayscale images. Then we build a features extraction model based on a siamese neural network that incorporates the VGG16 [19] network and efficient channel attention module (ECA) [20]. Finally, we utilize the features extracted by the network to perform clone detection.

B. Code Image Generation

The process of generating code images is illustrated in Figure 2. In this step, we cluster code fragments into different clone groups and record their corresponding indices. For code fragments of types T1, T2, ST3, and MT3, we group them based on the transitivity of the clones. Each clone pair forms a clone group for code fragments of types WT3/T4.

Next, we extract the decimal values of the ASCII encoding for each letter and symbol in the code snippet. For instance, the letter 'a' and the symbol '(' correspond to decimal ASCII values of 97 and 40, respectively. This process is repeated for all code fragments, generating a collection of ASCII sequences, each corresponding to a code fragment (as depicted in Figure 2). We then arrange these decimal values into a square matrix and convert it into a grayscale image. Each pixel in the resulting image represents a decimal value in the original sequence, with grayscale values ranging from 0 to 255.

To enable clone detection, we transform code snippets into grayscale images with a size of 105x105x1, which can accommodate up to 11025 pixels. This size is sufficient to represent most code snippets in the BCB and GCJ datasets. Although we could have used the standard input size of VGG16, 224x224, it would have included redundant information and slowed down network training. Therefore, we chose a smaller size of 105x105 to capture more effective semantic and structural features within an appropriate sequence length. We opted for a square image because DTL-DP [15] demonstrated that the network’s average prediction result improves when the image is closer to a square shape.
C. Network Construction

As depicted in Figure 3, we construct a siamese neural network with two VGG16 branches to extract features and use the code images generated in the previous step as inputs. After the two images pass through several convolution layers, maximum pooling layers, and RELU activation layers, they become two one-dimensional vectors, \( \delta_1 \) and \( \delta_2 \), each with a length of 4096. We then subtract the two one-dimensional vectors and calculate the L1-norm of interpolating the two eigenvectors, which is equivalent to finding the distance between the two vectors. Next, we perform two fully connected layers on this distance, with the second layer connected to a neuron whose result is passed through a sigmoid function to restrict the value between 0 and 1. This value represents the similarity between the two input code images.

In the VGG16 structure, we incorporate two attention mechanisms: self-attention and efficient channel attention (ECA). These mechanisms extract structural-semantic features of code images, assign weights, and highlight the differences between the two vectors. We will provide detailed descriptions of these modules later.

1) Network Backbone: We adopt VGG16 as our network’s backbone, consisting of five convolutional layers. Each convolutional layer comprises two 3x3 convolution operations and one 2x2 maximum pooling layer, and each feature layer produces 64, 128, 256, 512, and 512 channels, respectively. The input image size is 105x105x1, and the output is 4096x1.

2) Self-Attention Module: To improve the clone detection performance of VRL4CCD by allowing it to focus on key features in different images of code clone fragments, we introduce a self-attention mechanism into our model inspired by the outstanding performance of self-attention in GANs [21]. The self-attention mechanism can effectively capture long-range dependencies by computing the relationship between two locations of code images without stacking many convolutional layers to establish connections between each pixel in the image and other pixels.

As illustrated in the branch of the siamese neural network in Figure 3, we incorporate the attention mechanism into the last three convolutional layers of VGG16. In the attention layer, given a feature map \( x \), we use a 1x1 convolutional layer to linearly map the input features \( x \) resulting in \( f(x) \), \( g(x) \), and \( h(x) \), where \( f(x) = W_f x \), \( g(x) = W_g x \), and \( h(x) = W_h x \), and \( W_f \), \( W_g \), and \( W_h \) are learned weight matrices. For example, if the width, height, and number of channels of \( x \) are \( W \), \( H \), and \( C \), respectively, the size of \( x \) is \([C, N]\), where \( N = W \times H \), and the size of \( f(x) \) and \( g(x) \) is \([C/8, N]\). The transposed \( f(x) \) and \( g(x) \) are matrix-multiplied to obtain the auto-correlation in the features, i.e., the relationship of each pixel to all other pixels, where \( S_{ij} = f(x_i)^T g(x_j) \). We then apply the softmax activation function to obtain an attention map \( (\beta_{j,i}) \), which indicates the degree of attention the model pays to the \( i \)-th position when generating or obtaining the feature of the \( j \)-th pixel. Next, we multiply \( \beta_{j,i} \) pixel by pixel with \( h(x) \) to obtain adaptive attention feature maps \( o \). The output of this layer is \( o \).

\[
\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})}, o_j = \sum_{i=1}^{N} \beta_{j,i} h(x_i)
\]

where \( \exp \) is the logarithmic function, and \( o = (o_1, o_2, \ldots, o_j, \ldots, o_N) \in \mathbb{R}^{C \times N} \).

3) ECA Module: This module enhances the performance of cross-channel interaction. It allows the aggregation of convolutional features from multiple channels of code images to improve the code structure and semantic feature extraction. The module utilizes a one-dimensional convolution with a kernel size \( k \) that is adaptively determined and learns channel attention through the sigmoid function. As illustrated in Figure 3, the ECA module captures local cross-channel interactions by considering each channel and its \( k \) neighbors, which is:

\[
k = \psi(C) = \frac{\log_2 C + b}{\gamma} \bigg|_{\text{odd}}
\]

where \( C \) is the channel size, \( \gamma \) and \( b \) are the parameters of mapping function, and \( \text{odd} \) means to take an odd number.

In the ECA module, \( W_k \) represents the learned local cross-channel information interaction attention, which is:

\[
\begin{bmatrix}
  w^{1,1} & \cdots & w^{1,k} & 0 & 0 & \cdots & 0 \\
  0 & w^{2,2} & \cdots & w^{2,k+1} & 0 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & 0 & w^{c,c-k+1} & \cdots & w^{c,c}
\end{bmatrix}
\]

where \( W_k \) involves \( C \cdot k \) parameters.
To obtain the final output prediction value \( y \) and \( \delta \), two branches of the siamese neural network pass through two fully connected layers, we can obtain two high-dimensional vectors: \( \delta_1 \) and \( \delta_2 \). We subtract these two vectors and take the absolute value to obtain the final output prediction value \( y \). Therefore, for the \( i \)-th input feature map \( x_i \), the \( w_j \) and final output is given by:

\[
w_j = \sigma \left( \sum_{i=1}^{k} w_{ij} \sigma_{ij} \right), \quad y_i = \alpha w_i + x_i \tag{4}\]

where \( \Omega_i^j \) indicates the set of \( k \) adjacent channels of \( \sigma_j \). Where \( y_{ij} \in \Omega_i^j \), and the \( \alpha \) variable is a learnable variable that is initialized to 0. Introducing \( \alpha \) allows us to rely on nearby regions to provide informative cues and gradually assign more weight to non-local regions.

**IV. EXPERIMENT DESIGN**

**A. Datasets**

The BCB dataset is a well-known benchmark for code clone detection tasks, comprising over 6 million true clone pairs and 260,000 false clone pairs [8] from 10 functions, where each code instance represents a Java method. The GCJ dataset [22] is a collection of Java files from Google’s annual online programming competition, and we use the version curated by [12], which includes 1,669 Java files.

**B. Compared Methods**

We compare our approach VRL4CCD with the following code clone detection methods:

- **GGNN** [23] is a variation of graph neural networks that updates the node’s representation by incorporating information from neighboring nodes.
- **ASTNN** [24] utilizes recursive neural networks (RNN) to encode AST subtrees for statements, then feeds the encodings of all statement trees into an RNN to compute the vector representation for a program.
- **FA-AST** [12] enhances original ASTs by adding control and data flow edges, building a graph representation of programs called Flow-Augmented Abstract Syntax Tree (FA-AST), and applying two different types of graph neural networks on FA-AST to measure the similarity of code pairs.
- **TBCCD** [25] is a state-of-the-art code clone detector that uses ASTs and tree-based convolutions to measure code similarity.

**C. Evaluation Indicators**

To evaluate the detection performance, we utilize precision, recall, and F1-score as the evaluation metrics of our code clone detection research [26]. Specifically, if clones exist between two code fragments and the prediction result is a clone, it is referred to as a true positive.
(TP). Otherwise, it is considered a false positive (FP). Similarly, if two code fragments are not clones, and the prediction result is not a clone, it is called a true negative (TN). Otherwise, it is a false negative (FN). Then, precision (P), recall (R), and F1-score can be defined as:

\[ P = \frac{TP}{TP + FP} \]  
\[ R = \frac{TP}{TP + FN} \]  
\[ F1-score = \frac{2 \times P \times R}{P + R} \]

### Table I
RESULTS ON THE BCB AND GCJ DATASET

<table>
<thead>
<tr>
<th>Model</th>
<th>BCB</th>
<th>GCJ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>GGNN</td>
<td>0.72</td>
<td>0.89</td>
</tr>
<tr>
<td>ASTNN</td>
<td>0.92</td>
<td>0.94</td>
</tr>
<tr>
<td>FA-AST</td>
<td>0.96</td>
<td>0.94</td>
</tr>
<tr>
<td>TBCCD</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>VRL4CCD</td>
<td>0.98</td>
<td>0.98</td>
</tr>
</tbody>
</table>

### Table II
F1-score COMPARISON WITH VARIOUS CLONES TYPES IN BIGCLONEBENCH DATASET

<table>
<thead>
<tr>
<th>method</th>
<th>T1</th>
<th>T2</th>
<th>ST3</th>
<th>MT3</th>
<th>WT3/T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGNN</td>
<td>1.0</td>
<td>1.0</td>
<td>0.79</td>
<td>0.70</td>
<td>0.60</td>
</tr>
<tr>
<td>ASTNN</td>
<td>1.0</td>
<td>1.0</td>
<td>0.99</td>
<td>0.99</td>
<td>0.93</td>
</tr>
<tr>
<td>FA-AST</td>
<td>1.0</td>
<td>1.0</td>
<td>0.99</td>
<td>0.98</td>
<td>0.95</td>
</tr>
<tr>
<td>TBCCD</td>
<td>1.0</td>
<td>1.0</td>
<td>0.98</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>VRL4CCD</td>
<td>1.0</td>
<td>1.0</td>
<td>0.99</td>
<td>0.99</td>
<td>0.98</td>
</tr>
</tbody>
</table>

V. EXPERIMENT RESULT

Table I presents our code clone detection experiments’ precision, recall, and F1-score values. Since deep learning-based methods have inherent randomness, we executed each method 20 times and recorded their averages. The table shows that our VRL4CCD outperforms the other four methods in all metrics. Specifically, on the BCB dataset, VRL4CCD achieved the highest precision, recall, and F1-score values, all exceeding 0.98, surpassing all the comparison methods. Furthermore, the results of VRL4CCD are highly stable, as the precision, recall, and F1-score values are very close.

Detecting clones in the GCJ dataset is more challenging than in BCB. However, unexpectedly, VRL4CCD showed better in detecting clones in the GCJ dataset than in BCB. This reflects the superior ability of visualization methods to detect clones at the semantic level.

Furthermore, since the results in Table I are obtained by training the model on a mixture of all clone types, it indicates that VRL4CCD can detect a specific type of clone alone and function as a unified model to detect all types of clones. This demonstrates the strong generalization ability of VRL4CCD.

Table II presents the F1-score values of the compared methods on different types of clones. It is evident that our method exhibits a strong detection effect on T1, T2, and ST3 types of clones. Furthermore, the detection performance of VRL4CCD on MT3 and WT3/T4 types is comparable to the state-of-the-art clone detection methods in recent years. The results in Table II also highlight the superior performance of VRL4CCD on ST3, MT3, and WT3/T4 types of clones. Among all the methods, better detection results are observed for T1 and T2 types of clones. For the challenging MT3 and WT3/T4 types, VRL4CCD shows a 2 to 5 percent improvement compared to other baseline methods.

VI. DISCUSSION

A. Are VRL4CCD Suitable For Code Clone Detection?

Regarding the model structure, siamese neural networks are an excellent approach for performing image similarity discrimination tasks. Furthermore, we combined the siamese neural network with the VGG16 network and attention mechanism to enhance the model’s effectiveness. The experimental results demonstrate that the principle of image processing is suitable for processing code images, and VRL4CCD already demonstrates remarkable performance even when using default parameters.

In terms of experimental results, as shown in Table I, our method achieved high precision, recall rate, and F1-score values, with VRL4CCD even reaching 0.99 on the GCJ dataset. These results reflect the effectiveness of VRL4CCD.

In practice, we demonstrate a pair of real clones (Fig.4(a) and Fig.4(b)), where both code snippets achieve file copying. Our method, VRL4CCD, can accurately identify this clonal pair. Additionally, Fig.5 displays a pseudo-cloning pair in BCB, where Fig.5(a) performs the function of URL content crawling and Fig. 5(b) completes the function of file copying. Although these two code fragments are similar at the token and statement levels, our method can still recognize that they are not clones. These two examples demonstrate that our method, VRL4CCD, can effectively learn features from the training data to discriminate between code clones.
by mining the relationship between each character and pixel in the code.

Moreover, we improve the feature extraction process by incorporating a self-attention mechanism into the network. This allows the network to focus on the pixel-to-pixel associations, i.e., character-to-character associations in code snippets. We also introduce the ECA module to enhance the performance of cross-channel network interaction. We add the above two attention mechanisms to the bottleneck region of the network, which has the largest number of separated channels, further to improve the feature extraction capability of the network.

C. Threats to Validity

1) Implementation of Compared Methods: In our experiments, we implemented some baseline methods, such as TBCCD and ASTNN, using their open-source code available online. For the baseline methods without open-source code, we followed the details mentioned in the original paper as closely as possible to ensure their implementation.

2) Precision, Recall, and F1-Score Might Not be the Only Appropriate Measures: Although we used the most widely used metrics, i.e., precision, recall, and F1-score, to evaluate the effectiveness of code clone detection, other performance indicators could also be used, such as AUC, MCC, and G-mean.

3) Generalization of Experimental Results Might Be Limited: We conducted experiments on the BCB and GCJ datasets, which have different data scales, to enhance the generalization of our method. However, we cannot guarantee that VRL4CCD will achieve similar improvements on other datasets.

VII. CONCLUSION

This paper explores the application of software visualization techniques to code clone detection. Our proposed clone detection method, VRL4CCD, utilizes code visualization and a siamese neural network with a self-attention mechanism to extract code similarity-related features. Our experimental results demonstrate that VRL4CCD outperforms current state-of-the-art code clone detection methods. Moving forward, we plan to conduct more code detection tasks in real-world scenarios and explore further applications of software visualization technology in the area of code representation learning.

ACKNOWLEDGEMENT

This work is supported in part by the Guangdong Basic and Applied Basic Research Foundation (No. 2022A1515110564), in part by the Science and Technology Program of Guangzhou (No. 202201010312), in part by the Youth Innovative Talents Project of Ordinary Universities of Guangdong (No. 2020KQNCX008).

REFERENCES


Heterogeneous Directed Hypergraph Neural Network over abstract syntax tree (AST) for Code Classification

Guang Yang*, Tiancheng Jin*, Liang Dou*†

* School of Computer Science and Technology, East China Normal University, Shanghai, China
† NPPA Key Laboratory of Publishing Integration Development, ECNUP, Shanghai, China

51215901104@stu.ecnu.edu.cn, 51184506019@stu.ecnu.edu.cn, ldou@cs.ecnu.edu.cn

Abstract—Code classification is a difficult issue in program understanding and automatic coding. Due to the elusive syntax and complicated semantics in programs, most existing studies use techniques based on abstract syntax tree (AST) and graph neural network (GNN) to create code representations for code classification. These techniques utilize the structure and semantic information of the code, but they only take into account pairwise associations and neglect the high-order correlations that already exist between nodes in the AST, which may result in the loss of code structural information. On the other hand, while a general hypergraph can encode high-order data correlations, it is homogeneous and undirected which will result in a lack of semantic and structural information such as node types, edge types, and directions between child nodes and parent nodes when modeling AST. In this study, we propose to represent AST as a heterogeneous directed hypergraph (HDHG) and process the graph by heterogeneous directed hypergraph neural network (HDHGNN) for code classification. Our method improves code understanding and can represent high-order data correlations beyond paired interactions. We assess heterogeneous directed hypergraph neural network (HDHGNN) on public datasets of Python and Java programs. Our method outperforms previous AST-based and GNN-based methods, which demonstrates the capability of our model.

Index Terms—hypergraph, heterogeneous graph, code classification, graph neural networks, program understanding

I. INTRODUCTION

With the advancement of modern computer software, how to learn from vast open-source code repositories to enhance software development has become an essential research topic. In recent years, source code processing, which tries to help computers automatically comprehend and analyze source code, has received a lot of attention. Several works have been suggested including code classification [1]–[6], method name prediction [3] [4] [7] [8], code summarization [3] [9] [10] and code clone detection [5] [11] [12], etc.

Due to the improvement of machine learning technology, particularly deep learning, more and more work has employed deep learning for code classification. Currently, there are two main categories of code classification methods: AST-based and GNN-based. To take advantage of the semantic and structural information of the source code, several studies adopt AST when learning code representations [1] [5] [7] [8]. Some research uses graph neural networks (GNN) to create code representations for code categorization to a better understanding of the structure of code based on AST [3] [4] [13] [14].

Although these AST-based and graph-based techniques employ the structural information of source code and demonstrate their effectiveness, there is a problem that they only take into the pairwise relationships and ignore the possible high-order correlations between AST nodes. For example, when code is parsed into an AST, each parent AST node has child AST nodes belonging to various fields or called attributes. A parent node may have several child nodes under the same field, and these nodes have high-order correlations with one another. Fig. 1 depicts an example of a python code snippet. The corresponding AST generated by the official python ast module is illustrated in Fig. 2(a). As we can see, the “Module” is the root node of the AST. It has two child nodes, “Assign” and “Expr” which belong to the field named “body.” When modeling the correlations between the three nodes, previous approaches only consider the pairwise relationships, i.e., the pair of “Module” and “Assign” and the pair of “Module” and “Expr,” as demonstrated in Fig. 3(a). The high-order data correlation that “Assign” and “Expr” both belong to the “body” of “Module” as shown in Fig. 3(b) is dismissed may result in the loss of code structural information.

In recent years, hypergraph, which can encode high-order data correlations, has drawn a lot of interest. Considering the outstanding performance of hypergraph in graph classification [15] [16], we present hypergraph into code classification. On the other hand, a general hypergraph is homogeneous and undirected, i.e., it only has one type of node and one type of edge, and its hyperedge is undirected. If we represent the AST with a general hypergraph, it will result in lack of semantic

**Fig. 1.** An example of the code snippet. The program reads three inputs a, b, and c in turn, if a equals c, “Yes” will be output, otherwise, “No” will be output.

**Fig. 2.** A representation of the code snippet as an AST. Each node is a representation of a code. Each edge represents the interaction between adjacent code. For instance, “Assign” and “Expr” both belong to the “body” of “Module.”

**Fig. 3.** A representation of the code snippet as a hypergraph. Each edge represents the interaction between all code. For instance, “Assign” and “Expr” both belong to the “body” of “Module.”

DOI reference number: 10.18293/SEKE2023-136

1https://docs.python.org/3/library/ast.html.
II. RELATED WORK

Code classification is to classify codes based on their functions. Different from natural language, code has structural information. As a result, several works adopt AST by various techniques. Mou et al. [1] is one of the first works to suggest a Tree-Based Convolutional Neural Network (TBCNN) in code classification. Alon et al. propose code2seq [7] and code2vec [8] to deconstruct code to a collection of paths in its AST. J. Zhang et al. [5] propose a novel neural called ASTNN for source code representation for code classification and clone detection. N. D. Q. Bui et al. [18] propose a novel method named TreeCaps by fusing capsule networks with TBCNN in code classification.

With the popularity of GNN, more works apply kinds of GNN in code classification based on AST to strengthen the comprehension of code structures. M. Allamanis et al. [13] first construct graphs from source code by adding edges like control flow and data flow to AST and employing a gated graph neural network (GGNN) to process program graphs. V. Hellendoorn et al. [19] propose a model called GREAT based

III. PRELIMINARY

In this section, we introduce some fundamental background ideas, such as hypergraph, heterogeneous graph, and AST.

A. Hypergraph

In an ordinary graph, an edge can only be connected with two vertices. Different from general graphs, the edge of a hypergraph [20] can link any number of vertices. Formally, a hypergraph $H$ is a pair $H = (V, E)$ where $V$ is a set of elements called nodes or vertices, and $E$ is a set of non-empty subsets of $V$ called hyperedges or links.

A directed hypergraph [21] is a hypergraph with directed hyperedges. A directed hyperedge or hyperarc is an ordered pair, $E = (X, Y)$, of (possibly empty) disjoint subsets of vertices; $X$ is the tail of $E$ while $Y$ is its head. A backward hyperarc, or simply B-arc, is a hyperarc $E = (X, Y)$ with $|Y| = 1$. A forward hyperarc, or simply F-arc, is a hyperarc $E = (X, Y)$ with $|X| = 1$. A hypergraph whose hyperarcs are B-arcs is known as a B-graph (or B-hypergraph). A hyperarc whose hyperarcs are F-arcs is known as an F-graph or F-hypergraph.

In our study, since the child node of AST points to the parent node and the child node has only one parent node, our HDHG is a B-hypergraph.

B. Heterogeneous Graph

A heterogeneous graph [22] is a graph consisting of multiple types of entities or nodes and multiple types of links or edges. A heterogeneous graph is represented as $G = (V, E)$ consisting of an entity set $V$ and a link set $E$. A heterogeneous graph is also correlated with a node type mapping function $\phi : V \rightarrow A$ and a link type mapping function $\psi : E \rightarrow R$. $A$ and $R$ represent the sets of predefined object types and link types, where $|A| + |R| > 2$.

C. Abstract Syntax Tree

The AST represents the source code's abstract syntax structure. The code compiler will parse the code into an AST through the program syntax and semantic analysis. Each node on the tree represents a structure in the source code and belongs to different AST node types. Each AST node has zero, one, or several fields that can be thought of as the node's attributes. Each field may have none, one, or a list of objects such as AST node, number, and string. If one AST node contains a field with a different AST node, and the latter is equivalent to the former’s child AST node.

IV. METHODOLOGY

We first convert the code snippet into an AST and construct an HDHG based on it, then put it into our HDHGN. We combine the vector representations for code categorization once we get the network’s node’s vector representation. The overview of our model is demonstrated in Fig. 5.

A. Heterogeneous Directed Hypergraph

We parse the code snippet into an AST with a code compiler, then we develop the HDHG based on the AST. We set the node of AST as the “AST” node and the identifier of AST as the “identifier” node in HDHG. We set the value of the “AST” node as its AST node type name, set the value of the “identifier” node as its content, and treat them as two different types of nodes. The field is configured as a directed hyperedge. If one node has a field including another node, the latter node belongs to the tail of the field hyperedge, the former is the head of the field hyperedge. We designated the field name as the type of hyper edge. The illustration of the HDHG of AST in Fig. 2 is shown in Fig. 6.

B. Heterogeneous Directed Hypergraph Neural Network

1) Definition: We let a HDHG $G = (N, E)$, which includes a node set $N = \{n_1, n_2, \ldots, n_{|N|}\}$ and a directed hyperedge set $E = \{e_1, e_2, \ldots, e_{|E|}\}$. Each node $n = (\mu, x)$, where $\mu$ represents node type and $x$ is the value of the node. Each directed hyperedge $e = (\rho, S(e), T(e))$, $\rho$ represents edge type, $S(e) = \{n_1, \ldots, n_{|S(e)|}\} \subseteq N$ is the tail nodes of hyperedge $e$, $T(e) \subseteq N$ is the head node of hyperedge $e$, they show the direction of the hyperedge $e$ is $S(e)$ to $T(e)$.

2) Feature initialization: According to the value $x$ and the category $\mu$ of node $n$, we obtain embedding vector $d_n \in \mathbb{R}^{C_1}$ by embedding function as (1), where $C_1$ is the dimension size of the embedding vector.

$$d_n = \text{Embed}_\mu(x) \quad (1)$$

To put embedding vectors of various types into the same vector space, we make a linear projection to obtain the initial feature vector $h_0^\mu \in \mathbb{R}^{C_2}$ of node $n$ based on the corresponding node type $\mu$ as (2), where $C_2$ is the dimension size of feature vector and hidden vector.

$$h_0^\mu = W_\mu d_n + b_\mu \quad (2)$$

We also obtained embedding vector $d_e \in \mathbb{R}^{C_2}$ of hyperedge $e$ according to the edge type $\rho$ as (3).

$$d_e = \text{Embed}_{\text{edge}}(\rho) \quad (3)$$

3) Heterogeneous Directed Hypergraph Convolution Layer: Our model updates each node vector in one heterogeneous directed hypergraph convolution (HDHGConv) layer. We refer to the framework of two-stage message passing of hypergraph neural network [15] which has two steps: aggregating messages from nodes to hyperedges and aggregating messages from hyperedges to nodes. Contrarily, we add operations that add heterogeneous information and direction information.
### Aggregating messages from nodes to hyperedges:

First, the hidden vector $h_n^{l-1}$ of each node $n$ is multiplied by the head matrix or tail matrix to get the message vector $m_{e_n}^l$ from node $n$ to hyperedge $e$ as (4), where $l = 1, 2, \ldots, L$ indicate layer number, $L$ is the total number of layers.

$$m_{e_n}^l = \begin{cases} W_{\text{head}}h_{n}^{l-1} + b_{\text{head}} & \text{if } n \in S(e) \\ W_{\text{tail}}h_{n}^{l-1} + b_{\text{tail}} & \text{if } n \in T(e) \end{cases}$$  

Directed hyperedge $e$ gathers message from their tail nodes and head node by transformer attention mechanism [23]. For each message, the attention score is formulated as (5), where $d_e$ is the edge type vector.

$$\alpha_{n,e}^l = \text{Softmax} \left( \frac{(W_{q_1}d_e^T)W_{k_1}m_{e_n}^l}{\sqrt{C_2}} \right)$$

We obtain the vector $o_e^l$ of directed hyperedge $e$ as (6).

$$o_e^l = \sum_{n \in S(e) \text{ or } n \in T(e)} \alpha_{n,e}^l W_{\text{en}}^l m_{e_n}^l$$

Then we add edge type vector to $o_e^l$ as (7), where $z_e^l$ is formed as (8).

$$q_e^l = o_e^l + z_e^l$$

$$z_e^l = W_z^l d_e + b_z^l$$

### Aggregating messages from hyperedges to nodes:

For each directed hyperedge $e$ whose head node or tail node is $n$, the $q_e^l$ will be linear projected by (9) to get message $m_{e_n}^l$ which will be sent to $n$.

$$m_{e_n}^l = \begin{cases} W_{\text{to head}}q_e^l + b_{\text{to head}} & \text{if } n \in S(e) \\ W_{\text{to tail}}q_e^l + b_{\text{to tail}} & \text{if } n \in T(e) \end{cases}$$

Same as before, we aggregate messages to get $v_n^l$ by the transformer attention mechanism.

$$\alpha_{e_n}^l = \text{Softmax} \left( \frac{(W_{q_2}h_{n}^{l-1})^T W_{k_2}m_{e_n}^l}{\sqrt{C_2}} \right)$$

$$v_n^l = \sum_{T(e)=n \text{ or } n \in S(e)} \alpha_{e_n}^l W_{\text{en}}^l m_{e_n}^l$$

Last, we update hidden vector $h_n^l$ of node $n$ by (12), where $\sigma$ is the elu activation function and GraphNorm is graph normalization [24].

$$h_n^l = \sigma \left( \text{GraphNorm} \left( W_{u_1}v_n^l + W_{u_2}h_{n}^{l-1} + b_u^l \right) \right)$$

The $W$ above are all the weight matrix, and the $b$ above are all bias vectors, which will be learned by training. All of the attention mechanisms mentioned above use multiple heads.

### C. Classification

When we obtain the node hidden vectors $h_1^L, h_2^L, \ldots, h_N^L$ from the last layer, we utilize attention pooling to aggregate the information of each node to obtain vector representation $r$ as (13)(14), where $g \in \mathbb{R}^{C_2}$ is a learnable vector.

$$\alpha_n = \text{Softmax} \left( g^T h_n^L \right)$$

$$r = \sum_{n \in N} \alpha_n h_n^L$$
To obtain the final classification prediction, we use an MLP, which is expressed as (15).

\[
pred = \text{Softmax}(\text{MLP}(r)) \tag{15}
\]

The attention mechanism above is also multi-head attention. We employ the standard cross-entropy loss function for the training.

V. Evaluation

We implement code by torch_geometric\(^2\). Our implementation is available on https://github.com/qiankunmu/HDHGN.

A. Datasets

We use Python800 and Java250 to train and assess our model. The two public datasets are from Project CodeNet \cite{17} which are obtained from downloading submissions from two online judge websites: AIZU Online Judge and AtCoder. The code snippets are classified by the problem. The statistics of the datasets are depicted in Table I. To be clear, the AST node type means the AST type such as “Module” and “Assign,” different from node types in HDHG, i.e., “AST” and “identifier.” The edge type means the field name or called attribute in AST. We randomly split the dataset into the training set, validation set, and test set by 6:2:2.

B. Baselines

We compare our model with AST-based and GNN-based techniques which acquire the best performance in code classification including TBCNN \cite{1}, TreeCaps \cite{18}, GGNN \cite{13}, GREAT \cite{19} and HPG+HGT \cite{4}. TBCNN used a tree-based convolutional neural network to extract features from AST. A model called TreeCaps combines TBCNN and capsule networks. By adding edges like control flow and data flow to AST, the gated graph neural network (GGNN) processes graphs from source code. GREAT is a model extracting global relational information from code graphs based on the transformer architecture. A technique known as HPG+HGT uses a heterogeneous graph transformer to describe code as a heterogeneous graph. We also trained a GCN \cite{25} and a GIN \cite{26} in an experiment to compare.


<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STATISTICS OF THE DATASETS</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Size</td>
</tr>
<tr>
<td>Labels</td>
</tr>
<tr>
<td>Avg. node</td>
</tr>
<tr>
<td>Avg. edge</td>
</tr>
<tr>
<td>AST node types</td>
</tr>
<tr>
<td>Edge types</td>
</tr>
<tr>
<td>Language</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACCURACY OF MODELS IN CODE CLASSIFICATION (IN %).</strong></td>
</tr>
<tr>
<td>Models</td>
</tr>
<tr>
<td>AST-based</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>GNN-based</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Ours</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACCURACY OF ABLATION STUDY ON PYTHON800 (IN %).</strong></td>
</tr>
<tr>
<td>Variant</td>
</tr>
<tr>
<td>HDHGN</td>
</tr>
<tr>
<td>- hyperedge</td>
</tr>
<tr>
<td>- heterogeneous information</td>
</tr>
<tr>
<td>- direction</td>
</tr>
</tbody>
</table>

C. Experiment settings

We use a parser from the official Python 3.8 ast library and javalang library\(^3\) to parse the code snippets into ASTs. The embedding vectors are produced by random initialization and learned via training. Our model’s layer number was set to four. The hidden vector dimension size and embedding vector dimension size were both set to 128. We use a narrow multi-head attention \cite{23} mechanism and set the number of heads to eight. We employed Adam optimizer with the learning rate of 5 \times 10^{-5} to train our model. We set the dropout rate to 0.2. We optimized the hyper-parameters of other baselines for the validation set’s greatest performance. The models were trained for 100 epochs and we saved the models which perform best in validation set.

D. Results

We use the performance of the model on the test set as the outcome. We select classification accuracy as the metric. We calculate the mean accuracy and standard deviation after five iterations of the experiment. The results are depicted in Table II. Our HDHGN outperforms other methods in both datasets. In Python800, our HDHGN is 2.88% higher than the best baseline. In Java250, our model outperforms baseline models by at least 2.47%. This demonstrates that our model utilizes the semantic and structural features of code AST more effectively than previous approaches.

E. Ablation study

We perform some ablation studies of our HDHGN on Python800. We take into account three variants as below.

\(^3\)https://pypi.org/project/javalang/

\(^4\)We use the outcomes that were reported in their research because the paper did not make their code publicly available.
1) - hyperedge: We eliminate hyperedges from our model, leaving only paired edges, or normal edges, in the graph. A few regular edges will develop from the initial hyperedge.

2) - heterogeneous information: We eliminate heterogeneous information from our model, which entails treating identifier nodes and AST nodes as a single type of node in the graph and eliminating the information about edge types.

3) - direction: We remove direction information in our model, this means that the hyperedge is not directed hyperedge, it does not differentiate the head nodes and tail nodes.

We also repeat the experiment five times and compute the mean accuracy and standard deviation. The outcomes are depicted in Table III. Removing hyperedge make the result decrease by 3.08%. This demonstrates that high-order data correlations between AST nodes in code are indeed useful for comprehending programs. The removal of heterogeneous information reduces the result by 2.64%. Heterogeneous information often contains a lot of semantic information, which is helpful for program understanding. Removing direction caused a drop of 2.38% on the result. The direction of the graph can help enhance the model and get structural information by indicating whether the nodes connected by hyperedges are parent nodes or child nodes. The above outcomes demonstrate that our model can obtain a better understanding of AST structure and acquire more precise results in code classification after considering high-order data correlations, heterogeneous information, and direction information.

VI. CONCLUSION

In this study, we propose an HDHGN for code classification. To possibly encode high-order data correlations between nodes in AST, we introduce the use of hypergraphs. Due to the general hypergraph being homogeneous and undirected which will result in a lack of semantic and structural information, we propose to represent AST as a heterogeneous directed hypergraph. We create an HDHGN accordingly to utilize high-order data correlation, heterogeneous information and direction information better than previous methods. We test our model using open Python and Java datasets, and we compare the results to the baselines developed using the SOTA AST and GNN. The experiment demonstrates that our HDHGN outperforms the baselines. Further ablation study describes that the HDHGN enhances the performance of code classification.

Presently, the hypergraph we produce is large and contains many nodes and edges. Future research will focus on ways to scale down hypergraphs for modeling AST and enhance the current hypergraph model to make it more effective at classifying codes.

ACKNOWLEDGMENT

This work was supported by the Open Research Fund of NPPA Key Laboratory of Publishing Integration Development, ECNUP.
Dispatching and Scheduling Dependent Tasks Based on Multi-agent Deep Reinforcement Learning

Shuaishuai Feng†, Xi Wu‡, Yongxin Zhao*†§, and Yongjian Li§
† Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
‡ The University of Sydney, Australia
§ State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China

Abstract—With the development of edge computing, a large number of tasks can be offloaded to the edge server for computing, among which the dispatching and scheduling of dependent tasks has attracted extensive attention. The offloading of dependent tasks mainly has the following problems: how to select an appropriate edge server for dispatching, how to arrange the scheduling order of edge servers to better schedule tasks, and how to solve the task dependency problem. In this paper, we propose a dispatching and scheduling method DAMD, based on reinforcement learning and multi-agent reinforcement learning, to solve the above three problems. Specifically, as the first step of DAMD, a reinforcement learning approach is designed to estimate the network load and dynamically dispatch tasks to the appropriate edge servers. Each edge server is regarded as an agent by a multi-agent reinforcement learning method, the second step of DAMD, which comprehensively considers the dependency relationship between tasks and the scheduling relationship between servers to achieve the efficiency and fairness of task scheduling. Finally, the results show that our method can better complete the task within the deadline and greatly reduce the average response time according to the time sensitivity requirement.

Index Terms—dependent task, deep reinforcement learning, edge computing, multi-agent deep reinforcement learning

I. INTRODUCTION

With the development of the Internet of Things (IoT), more and more applications can be processed on mobile terminals [1]. Limited by their limited (computing, storage, and bandwidth) capabilities, terminal devices may spend a lot of time performing the required tasks, potentially resulting in poor quality of service [2]. In order to obtain higher service quality and computing resources, we can offload tasks to remote cloud data centers, which have a large amount of computing resources but are far away from users [3]. However, the long distance between the cloud and the user also introduces significant communication latency, which is unacceptable for time-sensitive applications/services [4]. In this case, edge computing can play an important role [5]. In edge computing, many edge servers with computing resources are deployed close to users [6]. By offloading tasks from terminal devices to edge servers, users can get edge services with better quality of service (for example, lower latency and higher precision) than in the cloud computing model. But at the same time, resources on the edge server are limited and cannot provide services for all tasks, especially when there are a large number of tasks [7]. Therefore, the edge communication system needs to solve two basic problems: which edge server should be dispatched to accommodate the task and what order each edge server should schedule the task, namely task dispatching and scheduling problem [8]. In addition, task requests from the same user can usually be divided into a group of independent or dependent tasks, which are usually represented as a job. Among these dependent tasks belonging to a job, there is a strict execution sequence, and subsequent tasks need to wait for the completion of the previous task [9]. It is also a very important problem how to dispatch and schedule tasks under the guarantee of task dependency.

In the dispatch phase, the traditional method is to dispatch tasks to the nearest server [10], which may lead to unbalanced task allocation and overload of the server. meng et al. proposed a scheduling scheme based on reinforcement learning [11], but it does not consider the dependence between tasks. Sundar et al. considered dependent tasks [12], but they only considered one type of dependent tasks and the application was limited.

We propose a new task dispatching and scheduling method DAMD that combines reinforcement learning and multi-agent reinforcement learning. Specifically, we propose a dispatcher based on deep reinforcement learning and a scheduler based on multi-agent deep reinforcement learning. For the dispatcher, we take the deep Q-learning (DQN) approach and use the response time of the task as a reward to update the current state of the edge network conditions and server load in real time. This can effectively improve efficiency by selecting edge servers with maximum rewards, avoiding network congestion and server overload. For the scheduler, we adopt multi-agent deep reinforcement learning technology to fully consider the dependencies between tasks and the execution order of dependent tasks in other servers. As a result, each edge server can dynamically allocate resources based on the time-sensitive requirements of each task. Therefore, our approach minimizes the average task response time and maintains efficiency across

*Corresponding author: yxzha@sei.ecnu.edu.cn
DOI reference number: 10.18293/SEKE2023-059
all tasks. The main contributions of this paper are:
1) We apply reinforcement learning and multi-agent reinforcement learning to the dispatch and scheduling of dependent tasks, comprehensively considering the dependency relationship between tasks and the cooperation between edge servers. As far as we know, this is the first time to solve the dispatch and scheduling of dependent tasks in this scenario.
2) We abstract the edge server and task information, and consider the load and distance of the edge server, which has a good application in real life.

The rest of this paper is structured as follows. In Section II, we introduced relevant scenes and the proposal of problems. In Section III, we introduced system modeling and algorithm process in detail. In Section IV, we compared and analyzed the results. Finally, we summarize this thesis in Section V.

II. PROBLEM DEFINITION AND SYSTEM MODELING

A. Problem Definition

In this section, we introduce the basic assumptions and question formulation.

As shown in Figure 1, we consider a target network with edge servers, different edge nodes provide different computing power. For each edge server, there may be multiple applications configured. In this network, users send jobs from their terminal devices to the access points (AP), which then dispatch the jobs to the edge server of the target edge network. When the jobs arrives at the edge server, it waits in the task queue for processing. The jobs generated by users can be dispatched to different edge servers for calculation. A job contains one or more dependent tasks. For dependent tasks, subsequent tasks must wait for the previous tasks to complete, and a task can only be dispatched to one edge server. In this process, we only consider the dispatching of jobs from AP to server and the scheduling within the server.

The problem can be expressed as follows: at a certain time, a group of dependent tasks are dispatched to edge servers in multiple networks. Given edge server information and dependent task information, each task is dispatched to an edge server and the scheduling within the edge server, so that the tasks can be completed before the deadline as far as possible.

B. System Modeling

For the dispatching and scheduling of dependent tasks, we mainly consider the information of edge servers and tasks. The main components are described as follows:

Edge server: a group of server nodes \( N = \{N_1, N_2, \ldots, N_m\} \) deployed in a specific area, and each node has the following characteristics:
- \( N_i.CPU \): indicates the computing capacity of the server, that is, the CPU.
- \( N_i.bd \): indicates the bandwidth of the server.
- \( N_i.size \): indicates the memory size of the server.
- \( N_i.task(t) \): indicates the task running on the \( N_i \) server at time \( t \).
- \( N_i.wait(t) \): indicates the waiting queue on the \( N_i \) server at time \( t \).

Jobs: a set of tasks that accomplish a specific goal, with dependencies between tasks, is a Directed acyclic graph (DAG). Through DAG, we can obtain the dependency relationships between tasks
- Task: the smallest, indivisible task in a job. A task must be completed on one server.
  - \( T_i.size \): indicates the size of the task
  - \( T_i.start \): indicates the arrival time of the task
  - \( T_i.deadline \): indicates the end time of the task
  - \( T_i.pretasks \): indicates the set of previous tasks of the task, which means, the task can work only after the previous tasks of the task are completed.

To minimize task response time, we divide task delay into two parts: external delay and internal delay.

We divide the external delay \( D \) of a task into three parts:
- \( D_1 \): indicates the time for the task to arrive from the AP to the edge server.
- \( D_2 \): indicates the time when the edge server returns to the AP after the task is scheduled.
- \( D_3 \): indicates the time when the calculation result of the previous task of the task is returned from an edge server.

If the task is on the same edge server as the previous task, the time is 0.

Since the bandwidth of AP is generally large, for an arbitrary time \( t \), we assume that the link bandwidth of the task from AP to the edge server is determined by the bandwidth of the edge server, that is, the link bandwidth of the task to the edge server is the bandwidth of the edge server. In addition, there is a propagation delay \( k \) between AP and the edge server. Propagation delay \( k \) is related to the distance from the AP to the edge server. Then \( D_1 = k + T_i.size/N_i.bd \).

When the task is returned by the edge server after scheduling, we assume that the data volume of the scheduling result is small. In this process, only the propagation delay is considered, that is, \( D_2 = k \).

For dependent tasks, each task needs to wait for the result of the previous task. If the previous task of the task is not completed when it comes to the task scheduling, the task needs to wait for the transmission of the result before scheduling.
Due to the small amount of data processing results, $D_3 = k$; if all previous tasks have been completed when the task is scheduled, and the transmission time of previous task results is included in the waiting time of this task, then $D_3 = 0$.

Internal delay is mainly divided into two parts, task wait time and task processing time. When a task reaches the edge server, it will be placed in a waiting queue. When it comes to task scheduling, since the task is a dependent task, it is necessary to judge whether all the previous tasks of the task have been completed. The waiting time of the task is the completion time of the previous task in the queue minus the arrival time of the task. The processing time of the task is $T_i \cdot \text{size} / N_i \cdot \text{CPU}$. The storage resources of each edge server are limited. Therefore, the memory of the edge server must be determined before dispatched a task to the edge server. Dispatched can be performed only when the task size is smaller than the remaining memory of the server.

III. DISPATCHING AND SCHEDULING ALGORITHM

A. Algorithm Framework

Our algorithm is mainly divided into two steps. First, we use reinforcement learning algorithm to dispatch tasks from AP to edge server. In this step, we do not consider the dependency between tasks, and select appropriate edge server for each task to dispatch. The action set is the edge server, which selects the action by evaluate network and gives the reward according to the response time of the task. The shorter the task response time, the smaller the load on the edge server. After one action is completed, the data is stored in the experience pool and periodically updated with the data in the experience pool. Then, multi-agent reinforcement learning is used to realize internal scheduling of edge servers. We regard each edge server as an agent and set an experience pool for each edge server. Each edge server selects one task for scheduling at a time and judges whether the previous task of the task is completed before scheduling. The total action set is the tasks selected by each edge server. All edge servers select a task and give a reward. After total action is completed, the data is stored in their own experience pool, and the parameters are updated regularly with the data in the experience pool. In this way, the influence of other server scheduling and the dependency between tasks are fully taken into account. The algorithm flow is shown in Figure 2.

B. Task Dispatching Method

A typical reinforcement learning model consists of states, actions, strategies and rewards. Agents learn by interacting with the environment, make actions and get corresponding rewards.

State: The agent interacts with the environment to obtain the current state and make corresponding decisions at the same time. When a task arrives, the agent will obtain various information of the current edge server.

Action: For any task arrived, the agent finds a suitable edge server to dispatch by observing the environment. Therefore, the scope of the action is the set of all edge servers, defined as:

$$a_t \in \{n_1, n_2, \ldots, n_m\} = a$$

Policy: Task dispatching policy reflects the mapping relationship between state and action. In DQN, we use neural network to generate action.

Reward: After observing the environmental state at time $t$, the agent makes corresponding actions by strategy, and gets the corresponding reward at time $t + 1$, here we define the reward as $e^{-T}$, $T$ is the response time of the task, which reflects the

Fig. 2. Dependent task dispatching and scheduling algorithm

![Diagram of dependent task dispatching and scheduling algorithm](image-url)
load on the edge server, and the shorter the response time, the less the load.

DQN designed two networks, the evaluation network and the target network, both of which initially had the same structure and parameter configuration. One is used to predict Q estimate (MainNet), one is used to predict Q reality (target), the targetQ of Q reality is calculated as:

$$\text{targetQ} = r + \gamma * \text{Qmax}(s', a', \theta)$$  \hspace{1cm} (2)

The loss is estimated by targetQ and Q, and the loss function generally adopts the mean square error loss:

$$\text{LOSS}(\theta) = E[(\text{targetQ} - Q(s, a, \theta))^2]$$  \hspace{1cm} (3)

Initialize MainNet and target, update the MainNet parameters according to the loss function, and target is fixed. After several iterations, all the MainNet parameters are copied to the target network, and so on. The targetQ is fixed in a period of time, which makes the algorithm update more stable.

Algorithm 1 Task Dispatching Algorithm Based On Deep Q-learning

**Input:** Task information and edge server information

**Output:** The edge server to which the task is dispatched

1. Initialize replay memory $D$ to capacity $N$
2. Initialize action-value function $Q$ with random weights $\theta$
3. Initialize target action-value function $Q$ with weights $\theta^* = \theta$
4. for episode $i = 1$ to $N$
5. Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
6. for each step $t$
7. With probability $\epsilon$ select a random action $a_t$
8. otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
9. Take action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
10. Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
11. Store transition $(s_t, a_t, r_t, s_{t+1})$ in $D$
12. Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
13. Set $y_j = r_j$ if episode terminates at step $j+1$
14. Perform a gradient descent step on $(y_j - Q(\phi_{j+1}, a_j; \theta))^2$ with respect to the network parameters $\theta$
15. Every $C$ steps reset $Q = Q$
16. end for
17. end for
18. end for

C. Task Scheduling Method

When tasks are scheduled within the server, the scheduling sequence within one server may affect other servers due to the dependency between tasks. Therefore, we consider using the Multi-agent DQN (MDQN) to schedule tasks within the server. We create a task queue for each server, and each server gets a reward after selecting a task from the task queue to schedule.

Algorithm 2 Task Scheduling Algorithm Based On MDQN

**Input:** Task queue and edge server information

**Output:** Scheduling tasks within the server

1. Initialize replay memory $D$ to capacity $N$
2. Initialize the evaluation network $w_e$ and the target network $w_t$
3. Update action policy
4. for episode $i = 1$ to $N$
5. Initializes the server queue information
6. for each step $t$
7. for each server $M$
8. Choose action $A$ according to action policy and $Q(S, A, w_e)$
9. Take action $A$, observe $R$ and $S'$
10. Store $e = (S, A, R, S')$
11. Sample random pair of $e$ from memory
12. Calculate target $y = R + \beta \text{maxQ}(S', A', w_t)$
13. Train parameter $w_e$ with a gradient descent step $(y - Q(S, A, w_e))^2$
14. if update = true then
15. $w_t = w_e$
16. end if
17. $S \leftarrow S'$
18. end for
19. Update server queue information
20. end for
21. end for

State value function: The state value function is the expectation of the action value function about the action, and the action performed by the agent depends on the strategy function, so the state value function $V_t$ of the agent also depends on the strategy of all the agents. For a single agent, its state space is the state information of a single server and the actions of other agents. The total state space is the state space of all agents.

Reward: Different relationships between multiple agents will result in different rewards given by the environment. If there is a cooperative relationship between multiple agents, the agents receive the same reward from the environment. If multiple agents are in a competitive relationship, positive reward for one agent from the environment will lead to negative reward for another agent. In the task scheduling environment, we regard multiple agents as cooperative relationships, and give a unified reward after all servers select a task for scheduling.

Action value function: For each server, its action is to select a task from the task queue to schedule. We create an experience pool for each agent, and store these experiences in an experience pool during the early stages of training, when the agent interacts with the environment to make an action. The experience contains $S$, $R$, $A$, and $S'$ information.

At the same time, we set up a target network with the same structure as the evaluation network for training, because a single network design will make the algorithm fall into the feedback loop between the target and the estimated Q value.
and become unstable. Therefore, a target network is used to avoid estimates getting out of control. The parameters of the evaluation network were updated at each training step, and the evaluation network was used to estimate the Q value of the action. The parameters of the target network are relatively stable. After several steps, the target network updates the parameters to the same as the evaluation network.

IV. EXPERIMENTAL RESULTS

In this section, we will evaluate the performance of the proposed approach by generating simulated environment information and data information compared to the baseline approach.

A. Experimental Environments

We use data sets generated by the network, which include information such as arrival time, processing time, data size, etc. At the same time, there is a dependency relationship between tasks. We randomly generate the dependency graph between tasks, and use the network to generate simulated edge server information, including the memory size and CPU of the edge server.

B. Baseline Method Comparison

In order to better evaluate the performance of the method and reflect the efficiency and fairness of the method in task dispatching and scheduling, we conducted comparative experiments with the following three baseline algorithms.

First is dispatching method baseline, this section is the method used by the task to get from the access point (AP) to the edge server. In order to show the performance of our dispatching method, we compare it with the three baseline dispatching methods.

1) Nearest: Dispatch the task to the nearest edge server.
2) Random: Dispatch tasks randomly to an edge server.
3) Least load: Dispatch tasks to edge servers with minimal load, here we use the waiting time of the task to represent, the longer the waiting time of the task, the greater the load of the edge server.

Then is scheduling method baseline, this section describes the method used to schedule tasks internally at the edge server, which we compare with the following three methods.

1) First-Come-First-Serve: Arrange tasks according to the order in which they arrive. Early tasks are scheduled first and final tasks later.
2) Shortest-Job-First: Schedule tasks based on the completion time, the shorter the completion time, the earlier the task is scheduled.
3) Shortest-Deadline-First: Tasks are scheduled based on their deadline time. The earlier the deadline time, the more urgent the task is, and the earlier the task is scheduled.

Since tasks are scheduled in two parts, from the AP to the edge server and scheduled in the edge server, we combine a pair of dispatching and scheduling baselines to compare dispatching and scheduling performance. These are the Nearest + First-Come-First-Serve (NF), Random + Shortest-Job-First (RS) and Least load + Shortest-Deadline-First: (LS)

C. Evaluation Result

In this part, we will compare our method with the baseline method in the simulation environment, and verify the superiority of our method over the baseline method by comparing the response time of tasks and the deadline missing rate of tasks under the same environment.

1) The impact of the number of tasks.

We randomly generated the edge server information and the dependent task information to conduct the experiment, and the dependent task was dispatched and scheduled by these methods. By comparing the response time of the task and the deadline missing rate of the task, the performance of our method was compared with that of the baseline method. The result is shown in Figure 3, the vertical axis are average response time and deadline missing rate, the horizontal axis is number of tasks.

![Graph](image)

Fig. 3. The effect of different task numbers on average response time and deadline missing rate

When a task is dispatched by an AP to an edge server, the nearest edge server receives an extremely large number of tasks as only the task is dispatched to the nearest edge server, resulting in a very high average task response time and deadline missing rate. Our approach determines which edge servers to dispatch tasks to based on network conditions and the load on the edge servers, which results in good performance.

When a task is scheduled within an edge server, First-Come-First-Serve first schedules the task that arrives at the edge server first, which will cause the later task to wait longer, and if the later task is the previous task of the task in other edge servers, it will cause the dependent task in other edge servers to wait for a long time. The basic idea of Shortest-Job-First is to give higher priority to smaller tasks, which results in longer waits for larger tasks that arrive first, and does not take into account the impact of dependent tasks in other edge servers. Shortest-Deadline-First also does not take into account dependencies between tasks, and does not take into account the effects of dependencies on other edge servers, which makes it less effective.

2) The impact of task density

Task arrival density represents the number of tasks arriving at the edge server per unit time, and the larger the number, the higher task arrival density. We measured the performance of the different approaches in terms of average task response time and deadline missing rate. Our method and Least Load can
dispatch tasks to edge servers with lower load, and thus better handle higher task arrival densities. Moreover, our method can take into account the dependencies between tasks and the effects of other server scheduling tasks, so the performance is the best. The result is shown in Figure 4, the vertical axis are average response time and deadline missing rate, the horizontal axis is task density.

3) The impact of the number of edge servers

We compare the performance of the different approaches with a different number of edge servers. We can see that when the number of servers is small, the deadline missing rate of tasks is very high. This is because the task is only dispatched to a few servers, which results in server overload. However, as the number of servers increases, the load of server decreases and the deadline missing rate decreases. Among all methods, our method has the best performance in terms of average task response time and deadline missing rate. The result is shown in Figure 5, the vertical axis are average response time and deadline missing rate, the horizontal axis is number of edge servers.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a method DAMD combining DQN and MDQN for dispatching and scheduling dependent tasks. It can estimate the network condition and server load, and make reasonable arrangements according to the timeliness requirements of tasks. At the same time, it considers the dependency between tasks comprehensively, and make different servers cooperate with each other. We conducted evaluation experiments on simulated environments, and the result shows that the proposed method can make optimal actions through continuous learning from experience, so it performs well in dispatching and scheduling of dependent tasks. However, the algorithm can still be improved. Considering that the dependent task can be represented as a DAG graph, perhaps the introduction of graph neural network can improve the efficiency, simultaneously considering the competition and cooperation methods of multi-agent reinforcement learning. We will continue to carry out research and make contributions to the dispatching and scheduling of dependent tasks.

VI. ACKNOWLEDGEMENT

This work is supported by National Key Research and Development Program (2019YFB2102600), National Natural Science Foundation of China (NSFC 62272165), the “Digital Silk Road” Shanghai International Joint Lab of Trustworthy Intelligent Software (Grant No. 22510750100), and Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

REFERENCES

GAT-Team: A Team Recommendation Model for Open-Source Software Projects

Qing Qi†, Jian Cao†*, Duo Wang†

*Corresponding author
†Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
Email: {qi_ng616, cao-jian, wd_doylle}@sjtu.edu.cn

Abstract—The active participation of contributors is key to the success of any open source software (OSS) projects. In the open source community, it can be found that some developers have participated together in multiple projects and also built social connections so that they form an independent team. Independent teams can be high-quality developer candidates for OSS projects. By modeling the developers and independent teams, we design approaches to recommend independent teams to OSS projects. We used the interaction data between developers to build the developers’ social connection network, on which independent teams can be discovered by utilizing the community discovery algorithm. Based on the graph attention network, we designed a team recommendation model GAT-team. The performance of the GAT-team is evaluated on a real-world dataset. GAT-team achieves much better results compared with the other algorithms in the experiments. We provide the data and the code at https://doi.org/10.5281/zenodo.4646651.

Index Terms—Open Source Software Project, Independent Team, Team Recommendation, Graph Attention Network

I. Introduction

Finding and attracting developers with relevant expertise and interest to contribute to open-source software (OSS) projects is crucial to their sustainable development [1]. However, developers often find it hard to obtain suitable resources from tremendous OSS projects. This largely discourages their participation in OSS projects. Therefore, many approaches have been proposed to recommend developers to OSS projects [2]–[5].

Software development is a sophisticated task which inherently requires teamwork. Researchers have proposed approaches for team formation, task assignment, and conflict management, collaboration mechanisms and information sharing [6]–[9]. Comparing with teamwork in traditional projects, an OSS project team is often composed of unknowns and volunteers [10], who are often distributed in different places. Therefore, improving the collaboration efficiency of virtual teams working on OSS projects is a more significant challenge [11].

In OSS communities, the same group of developers can be found having participated in several projects. If these developers have built social connections and cooperate with each other directly, they form an independent team. Compared with project teams that only focus on the tasks of the specific project, members of an independent team are more likely to share common interests and develop long-lasting cooperative relationships.

It can be assumed that independent teams have higher productivity so as to they can contribute significantly to the OSS projects. There are several reasons for it. Firstly, the social connections between team members are beneficial to their collaboration. Secondly, they have lower startup costs. Corbitt et al. [12] studied the amount of time spent at each of the four classic team development stages (forming, storming, norming and performing for virtual teams), which is 30%, 18%, 17%, and 35%, respectively. This indicates that stable and long-lasting teams are more efficient than temporary teams because the formation cost of a virtual team is considerably large.

As an empirical study, we identified independent teams based on the data from GitHub and compared their contribution rates with other contributors (the approach to identify independent teams will be introduced later). The contribution data provided by GitHub REST APIs is adopted to measure members’ contribution rates. We normalized the data by dividing them by the average contribution of all contributors in each repository. Then a student’s t-test is applied to compare the contribution rates of independent team members and other contributors. The conclusion is the contribution rate of independent team member is expected to be 2.57-2.60 times higher than the average with 95% confidence.

This observation motivates us to recommend independent teams to OSS projects. Currently, few insights are available on recommending development teams to open-source projects. The contributions of this paper are as follows:

1) We provide an approach to identify independent teams in OSS communities.
2) We design GAT-team, a graph attention network based algorithm to recommend independent teams to OSS projects efficiently.
3) We perform extensive experiments on the real-world dataset and prove that GAT-team is significantly better than other models.

The paper is organized as follows. Section 2 reviews the related work. Section 3 presents an approach to discover independent teams in OSS communities. Section 4 introduces GAT-team, a team recommendation approach.
for OSS projects. Experiments are reported in Section 5. Finally, we conclude the paper in Section 6.

II. Related Work

A. Teamwork in Open Source Software Projects

Researchers are interested in enhancing the efficiency of teamwork for OSS development. For example, in [13], the influence of developers’ different involvements is investigated. Grottke et al. study the relationships between team factors and the efficiency of failure processing [14]. In [8], the mechanism of conflict management is proposed.

It is well known that relationship building and group cohesion are very important for virtual teams to be effective in accomplishing tasks. Research shows that developers are more likely to join projects that are initiated by those with whom they have socially connected in the past. In addition, social networks are the most common approach for finding cooperative groups [15]. McDonald found two social networks in an organization [16]. One is called the work group graph (WGG) which reflects shared work contexts. The other is based on the successive pile sort (SPS) which reflects workplace sociability. These two approaches to construct social networks are both adopted in our research. Moreover, Bird et al. [17] prove that the organizational structure in OSS projects can be reflected by the social connections among project participants. We therefore detect teams based on the resulting social networks. Our research is dedicated to recommending stable and context-independent developer teams.

B. Member Recommendation for Open Source Software Projects

Developers can be recommended to a project based on their expertise or past experience [2]. Many past research studies have focused on recommendation problems on GitHub [3]–[5]. A common approach is leveraging the interaction data between the developer and projects and the similarity data between projects to find projects similar to the ones on which the developer has already participated. In the work by Xu et al., it extracts TF-IDF features from the projects’ text files to compute the similarity between projects. After this, the Create, Fork, and Star actions of the developers are mined, weights to these actions are assigned, and the interaction matrix between developers and projects is obtained [3]. Other works try to solve the problem with neural network models. For example, Zhou et al. point out that due to the severe sparsity of recommendation problems on GitHub, the tradition collaborative filtering cannot achieve satisfying results. From this observation, they propose the Hierarchical Collaborative Embedding Model, which compensates for the data sparsity by constructing a knowledge network among projects[4].

In this study, we will recommend independent teams rather than individual members because members of independent teams are more productive than individual members.

C. Group Recommendation

Team recommendation is also related to the group recommendation problem. The difference is group recommendation approaches try to recommend items to a group while team recommendation approaches try to recommend teams to a project. However, they share a similarity in that potential connections between a group (or a team) and an item (or a project) are predicted.

A traditional approach for the group recommendation problem is to use a predefined consensus function to aggregate the recommendation results for each member into the results for the group. Garrett et al. use this tradition approach by combining the social connections between users into the consensus function, which achieves increased performance [18]. Recently, many works use the attention mechanism in group recommendation problems. The MoSAN model designed by Vinh Tran et al. assigns a child attention network for each member in the group and learns the preference of the members using the attention mechanism [19]. The work by Cao et al. combines neural collaborative filtering and the attention mechanism. The social connections between group members are also exploited to improve the performance [20]. Their work proves that the attention mechanism is superior to the several traditional consensus functions used in group recommendation such as Least Misery and Maximum Satisfaction.

III. Independent Teams in OSS Communities: Definition and Detection Approach

An independent team is a virtual team in OSS communities that has worked on more than one project and their members are socially connected. Developers on OSS platforms can have different types of social connections such as following each other, contributing to the same OSS project, and discussing in issue resolution processes. However, these connections are context-dependent and they do not necessarily indicate mutual connections between developers. For example, following can be uni-directional. Contributors under the same repository do not necessarily know each other, even if they have has jointly participated into the resolution process of the same issue. At the same time, developers can contact with each other beyond open source platforms, such as in google group or through emails. These communication channels may provide direct proofs that they are socially connected. Unfortunately, to obtain this data is not easy and has privacy issues.

Fortunately, there are other cooperation mechanisms provided by the open source platforms that can be directly adopted as an evidence, for example, mention action on GitHub. When you mention @ a GitHub username in the context of an issue or pull request, that person is notified and subscribed to future updates. When two developers
have mentioned each other, we can assume they have set up social connections. Therefore, our model relies on mention to detect social connections.

We can identify independent teams from OSS communities according to the definition. As the first step, we construct a developer social network (DSN). In this network: (1) Each node represents a developer. (2) For any two nodes, there is an edge linking them if and only if: a) the two developers have contacted with each other in some ways; b) the two developers have jointly contributed to multiple projects.

After constructing the DSN, we use community detection algorithms to detect teams with strong intra-connections and weak inter-connections in the network. Since a developer can belong to multiple independent teams, we choose the Order Statistics Local Optimization Method (OSLOM) to detect independent teams that are represented by overlapping communities [21].

The information used in the recommendation of independent teams includes: (1) Interest vectors of independent teams and developers; feature vectors of project repositories. (2) Project repositories to which each team or developer contributes to and their contribution values in each of the project repositories. (3) Teams to which each developer belongs, contribution rate and structural information (degree in the team sub-network of the DSN) of each member in each of the teams. (4) Social connections among developers (the DSN).

The feature vectors of project repositories include numeric features (size, forks, watchers and subscribers) and non-numeric features (language, topics). The interest vector of a developer is an aggregation of the feature vectors of all the project repositories on which the developer has worked. The contribution value of an independent team in a hosting project repository is the sum of the contribution values of all its members. The contribution rate of a team member is represented by the percentage of contributions the member made to the hosting project repositories of the team. The structural information of a team member is represented by its degree in the corresponding sub-network of the whole DSN.

IV. GAT-team: A GAT-based Team Recommendation Model

A. Model Design

Since the participation and collaboration of developers in OSS communities can be directly modeled as a graph, we try to make use of the graph neural network (GNN) to support team recommendation. Graph attention (GAT) network introduces attention mechanisms to the propagation step, which compute the hidden representations of each node in the graph, by attending over its neighbors, following a self-attention strategy. GAT is efficient because the operation is parallelizable across node-neighbor pairs. In addition, it can be applied to graph nodes having different degrees [22]. In order to thoroughly mine the relational information in OSS communities to support team recommendation, we designed a GAT-based team recommendation model (GAT-team). The architecture of the model is shown in Fig. 1.

The input features (Repo Embedding, Core Developer Embedding, Neighbor Developer Embedding and Team Embedding) consist of numeric features and Word2Vec embedding. For the three graph attention layers in the model, the input graphs are connections between the project repository and core developers, connections between core developers and their immediate neighbors, and connections between developers and teams.

Multi-head graph attention layers are used as basic units of GAT-team. In every multi-head graph attention layer, multiple parallel graph attention operations are computed on the input graph and the input embedding.

The outputs of the parallel graph attention units are aggregated and fed into the output attention unit to compute the final output embedding. For multi-head graph attention layers, common aggregation methods include averaging and concatenation. Our model uses concatenation. Finally, the output embedding of the multi-head graph attention layer is computed.

B. Propagation Mechanism

The first multi-head graph attention layer is responsible for the weight propagation between project repositories and core developers. The input graph contains the connections between the project repository and its core developers. It is necessary to mention that because a self-connection is attached to each node on the repository-developer graph, we compute self-attention for all the neighbor developers for the sake of consistency between the output embedding of core developers and neighbor developers. The second multi-head graph attention layer is responsible for the weight propagation between developers. The input graph is the DSN, which models the social connections between developers. And the output of inter-developer propagation is computed. The third multi-head graph attention layer is responsible for the weight propagation between developers and teams. The input graph is the connections between developers and teams. The output is the developer-team propagation.
C. Output Layer

The final module of the model consists of a feed forward network and a Sigmoid activation. The output of the model is computed by Equation (1):

$$
\hat{r}_{RT} = \sigma (A_{ff} P_T^* + b_{ff})
$$

where $A_{ff}$ and $b_{ff}$ are the learnable parameters of the model. We choose the binary cross entropy-loss as the loss function of GAT-team.

V. Experiments

A. Data Set

Our experiment is based on the data set of all the issue comment events on GitHub from 01/01/2015 to 06/30/2020. We also collect the information of 312,934 repositories and 1,384,736 users. The DSN is constructed according to the definition. The network has 719,202 nodes and 1,121,778 edges in total. A total number of 22,875 independent teams are found in the network.

B. Metrics

The discounted cumulative gain (DCG) measures the quality of the results in a ranked list, where items in that list are graded in some way. The grading is a relevance judgment for each result. DCG accumulated at a particular rank position $p$ is defined in Equation (2):

$$
DCG_p = \sum_{i=1}^{p} \frac{rel_i - 1}{\log_2(i + 1)}
$$

where $rel_i$ is the relevance score of $i$th element.

By computing the DCG value of the optimal recommendation result, one gets the ideal discounted cumulative gain (IDCG) of a single query. Normalized Discounted Cumulative Gain (nDCG) is computed by dividing the DCG by the IDCG. The higher the nDCG, the better ranked list.

C. Comparison Approaches

We developed a series of algorithms to compare them with GAT-team. Moreover, we select a group recommendation approach SoAGREE [20] for comparison.

1) Interest-Vector-Based Recommendation: An intuitive approach is to compute the similarity between the feature vectors of the teams and the project repositories. Since the connections between developers and teams are known, group recommendation techniques can be employed to improve the performance.

Interest Matching: For each team in the dataset, we calculate their suitability to project repositories using the following approach. For an interest vector of a team ($V_T$) and a feature vector of a project repository ($V_R$): (1) For numeric features, calculate the Euclidean distance between the two vectors. (2) For non-numeric features $A$ and $B$, calculate their Jaccard distance. (3) Calculate the distance between the two vectors by using the two aforementioned distances. (4) Take the reciprocal of the distance to get the similarity of the two vectors.

For each project repository, we sort the candidate teams in descending order of their suitability, and then take the first-$k$ teams as the recommendation result.

Group Aggregation: In group recommendation problems, the interest of the team is an aggregate of interests of all members. Using the known information in the dataset, we designed the following aggregation methods: (1) Interest Aggregation: Aggregate the interest vectors of the members into the interest vector of the team. Calculate the similarity between the team’s interest vector and the project repository’s feature vector. (2) Mean Aggregation: After calculating the suitability of all the members, take the mean value as the suitability of the team. (3) Least Misery: After calculating the suitability of all the members, take the minimum as the suitability of the team. (4) Maximum Satisfaction: After calculating the suitability of all members, take the maximum as the suitability of the team. (5) Expertise-Based Aggregation: Take a weighted sum as the suitability of the team. We adopted two strategies to weight each member, i.e., weight assignment based on contribution rate and weight assignment based on members’ degree in the team social network.

The recommendation algorithms designed based on the aforementioned aggregation methods are referred to as sim_interest, sim_mean, sim_lm, sim_ms, sim_exp_contri, and sim_exp_degree respectively in the experiments.

2) Core-Developer-Based Social Recommendation: Social connections between team members are essential information in team recommendation problems. From the aforementioned observations, we designed the Core-Developer-Based Social Recommendation algorithm: (1) For each repository, obtain a list of its core developers. (2) For each candidate team, obtain a list of its members. Count the number of social connections between team members and core developers of the repository. Take this number as the strength of the connection between the team and the repository. (3) Sort the candidate teams in the descending order of their connection strength with the repository. Take the first-$k$ teams in the sequence as the recommendation results.

The core-developer-based social recommendation is referred to as Social in the experiments.

3) SoAGREE: We also compared GAT-Team with the recommendation model based on social connections and graph attention networks: Social-Enhanced Attentive Group Recommendation (SoAGREE) [20]. We used the implementation shared by the authors and adjusted the hyperparameters to fit the dataset used in this study.

D. Ablation Study

In order to verify the architecture of GAT-Team, we employed the ablation study method. We designed the
following two alternations of the original model to verify the validity of the three-layer structure:

(1) ALT1: Combine the Repo-Developer layer and the Inter-Developer layer. The data inputs to the new GAT layer are Repo Embedding and Developer Embedding. The input graph to the new GAT layer is a combination of the Repo-Developer graph and Inter-Developer graph.

(2) Combine the Repo-Developer layer, the Inter-Developer layer, and the Team-Developer layer. The data inputs to the new GAT layer are Repo Embedding, Developer Embedding, and Team Embedding. The graph inputted to the new GAT layer is a combination of the Repo-Developer graph, Inter-Developer graph, and Team-Developer graph.

The two alternatives are referred to as GAT-team_alt1 and GAT-team_alt2 in the experiments.

In summary, we implemented 11 different recommendation algorithms in total (Interest_team, Interest_mean, Interest_ln, Interest_ms, Interest_exp_contri, Interest_exp_degree, Social, GAT-team, GAT-team_alt1, GAT-team_alt2, SoAGREE). We tested all the algorithms on the test set and used nDCG as the evaluation metric.

E. Experiment Results

The results of the experiments are shown in Fig.2. Figures 2(a) through 2(k) show the distribution of nDCG values achieved by all the algorithms when the value of $k$ is 30. Fig. 2(l) and Table I show the relationship between the average nDCG value achieved by the algorithms and the value of $k$. Table II shows the training errors and test errors of GAT-team, GAT-team_alt1, GAT-team_alt2, and SoAGREE.

As one can see from the experimental results, the GAT-team and the Core-Developer-Based Social Recommendation algorithm designed in our study achieve significantly better performance than other algorithms. For GAT-based models, the original architecture performs better than the two alternative architectures. While the performance of GAT-team is better than Social, GAT-team_alt2 is significantly out-performed by Social. Although SoAGREE achieves better performance than interest-vector-based recommendation algorithms, it is outperformed by the other algorithms by a large margin.

The success of GAT-team mainly lies in two aspects. Firstly, it effectively utilizes the information in a comprehensively way. Secondly, it thoroughly exploits the social connections with weight propagation and graph attention operations on the developer social network.

The SoAGREE model proposed by Cao et al. achieves outstanding performance on the MaFengWo dataset and CAMRa2011 [20], but it does not perform well in the experiments in our study. The reason is SoAGREE learns group embeddings only by aggregating the embeddings of users while neglecting more comprehensive information in OSS projects.

VI. Conclusion

This study focuses on how to recommend independent teams to OSS projects. For the first time, we suggest that independent teams can be high quality developer candidates for OSS projects. We designed and implemented a GAT-based recommendation model and compared it with the Interest-Vector-Based Recommendation algorithm, Core-Developer-Based Social Recommendation algorithm and SoAGREE, a recommendation model which works well on traveling information websites and movie rating websites. Our GAT-based model and the Core-Developer-Based Social Recommendation both achieve satisfying performance in the experiments, out-performing SoAGREE by a large margin. Additionally, we conducted an ablation study on the GAT-based model and verified the validity of our architecture.

References

(a) Interest_team
(b) Interest_mean
(c) Interest_lm
(d) Interest_ms
(e) Interest_exp_contri
(f) Interest_exp_degree
(g) Social
(h) GAT-Team
(i) GAT-Team_alt1
(j) GAT-Team_alt2
(k) SoAGREE
(l) Comparisons

Fig. 2. Experimental Results

TABLE I
Experimental Results

<table>
<thead>
<tr>
<th>Algorithm/Model</th>
<th>avg. ndcg@k=5</th>
<th>avg. ndcg@k=10</th>
<th>avg. ndcg@k=30</th>
<th>avg. ndcg@k=50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interest_team</td>
<td>0.04306</td>
<td>0.04986</td>
<td>0.05987</td>
<td>0.06415</td>
</tr>
<tr>
<td>Interest_mean</td>
<td>0.02597</td>
<td>0.03182</td>
<td>0.04187</td>
<td>0.04699</td>
</tr>
<tr>
<td>Interest_lm</td>
<td>0.02319</td>
<td>0.02805</td>
<td>0.03657</td>
<td>0.04069</td>
</tr>
<tr>
<td>Interest_ms</td>
<td>0.01065</td>
<td>0.01325</td>
<td>0.01688</td>
<td>0.01887</td>
</tr>
<tr>
<td>Interest_contri</td>
<td>0.00678</td>
<td>0.00836</td>
<td>0.01096</td>
<td>0.01208</td>
</tr>
<tr>
<td>Interest_degree</td>
<td>0.01184</td>
<td>0.01433</td>
<td>0.01847</td>
<td>0.02066</td>
</tr>
<tr>
<td>Social</td>
<td>0.63339</td>
<td>0.68949</td>
<td>0.73760</td>
<td>0.74919</td>
</tr>
<tr>
<td>SoAGREE</td>
<td>0.02511</td>
<td>0.03466</td>
<td>0.05931</td>
<td>0.07837</td>
</tr>
<tr>
<td>GAT-team</td>
<td>0.82743</td>
<td>0.86948</td>
<td>0.90172</td>
<td>0.90812</td>
</tr>
<tr>
<td>GAT-team_alt1</td>
<td>0.66080</td>
<td>0.71542</td>
<td>0.76180</td>
<td>0.77193</td>
</tr>
<tr>
<td>GAT-team_alt2</td>
<td>0.44937</td>
<td>0.52811</td>
<td>0.59278</td>
<td>0.60764</td>
</tr>
</tbody>
</table>

TABLE II
Training and Testing Error

<table>
<thead>
<tr>
<th>Model</th>
<th>training error</th>
<th>test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAT-team</td>
<td>0.01431</td>
<td>0.01966</td>
</tr>
<tr>
<td>GAT-team_alt1</td>
<td>0.02733</td>
<td>0.03712</td>
</tr>
<tr>
<td>GAT-team_alt2</td>
<td>0.04592</td>
<td>0.07248</td>
</tr>
</tbody>
</table>


An Empirical Study of Adversarial Training in Code Comment Generation

Yiheng Shen
Nantong University
China
yiheng.s@outlook.com

Xiaolin Ju*
Nantong University
China
ju.xl@ntu.edu.cn

Xiang Chen*
Nantong University
China
xchencs@ntu.edu.cn

Guang Yang
Nanjing University of Aeronautics and Astronautics, China
novelyg@outlook.com

Abstract—The code comment generation task is designed for developers to understand programs more quickly during development and maintenance. However, the existing automatic code comment generation models can not generate valuable comments for developers. It is necessary to explore a technology that can optimize the performance of code comment generation models without changing the model. We consider adversarial training as the experimental object, which can improve the robustness and generalization of the model. We present a large-scale study to experimentally validate the performance of gradient-based adversarial training methods in the code comment generation task. The results show that adversarial training can improve the model performance by generating adversarial examples without changing the model. Our empirical study can provide a new perspective for researchers to improve the performance of code comment generation models.

Index Terms—Code comment generation, Adversarial Training, Bash code, Deep Learning, Empirical Study

I. INTRODUCTION

Research shows that software developers and maintainers spend 59% [1] of their time on program understanding. A good code comment can improve the efficiency of software development and maintenance. Automating the generation of comments for code is a pressing issue. With the development of deep learning techniques and pre-trained models, the task of automatic code comment generation has also achieved SOTA (state-of-the-art) in academic research [2], [3]. However, recent empirical studies have shown that comments automatically generated for code based on existing models [3], [4] are ineffective in guiding developers. For example, Mastropaolo et al. [5] found that only 10% of the comments could reach the level of human writing. This suggests that software comment generation is still a development away from full industrial use.

For this reason, we consider using some techniques to optimize the model performance without changing the model, and adversarial training is a good way. Some studies used rule-based adversarial training methods in NLP (Natural Language Processing) tasks. For example, Ribeiro et al. [6] found that adversarial attacks can be useful when debugging NLP models. Zhang et al. [2] also found that model performance can be significantly improved in the field of code comment generation by generating adversarial examples. Different adversarial rules have been designed for various natural language processing (NLP) tasks. In this context, we focus on gradient-based adversarial training methods [8]–[11] and present a large-scale study evaluating the impact of adversarial training on code comment generation. Previous studies have not investigated whether gradient-based adversarial training methods can optimize the performance of code comment generation models.

In our empirical study, we explore the performance of gradient-based adversarial training methods on different code annotation generation models and different adversarial training methods on the same model. We considered four classic adversarial training methods: FGSM [8], FGM [9], PGD [10], and FreeLB [11]. Moreover, we also consider three types of code comment generation methods: deep learning-based [12], [13], pre-trained model-based [3], [4], [14], and hybrid model [5], [15], [16].

In summary, the contributions of our empirical study can be summarized as follows.

- To the best of our knowledge, we are the first to investigate improving the performance of code comment generation models by adversarial training.
- We conducted a large-scale empirical study to investigate this issue. In our empirical study, we selected five adversarial training methods and three types of code comment generation models. Our empirical results validate that adversarial training can improve the performance of code comment generation models. Moreover, we find that the PGD method is optimum for evaluating different adversarial training methods.
- To let other researchers follow our research, we shared our scripts, datasets, and results on the project homepage.

II. RELATED WORK AND RESEARCH MOTIVATION

In this section, we first analyze the work related to the task of adversarial training and code comment generation. After analyzing the relevant research, we emphasize the novelty of this research.

A. Code Comment Generation

Code comment generation [2], [3], [12] can be defined as a code understanding and neural machine translation problem.


* Corresponding author.

1https://github.com/syhstudy/AT_Empirical_Study
We classify these studies into information retrieval-based, deep learning-based, and hybrid methods. The method based on information retrieval is applicable to code with high reusability. Wong et al. [17] used SIM (a token-based code cloning detection tool) to detect stack overflow code fragments and corresponding descriptions, which used the detected pseudocode as the final comment. Recently, Yang et al. [18] proposed the method CCGIR, which retrieves smart contract codes based on semantic similarity, lexical similarity, and syntactic similarity. In terms of deep learning-based methods, Allamanis et al. [19] put forward a model, which uses CNN and attention mechanism to detect the attention characteristics of local time-invariant and remote topics, and uses GRU to decode the output. Recently, Yang et al. [22] proposed a new Transformer-based method CodeBERT and a fusion method based on mixed code representation. The hybrid method takes advantage of both information retrieval and deep learning. Li et al. [21] combined the code comments obtained by information retrieval with the semantic information of the input code to generate code comments. Recently, Yu et al. [16] proposed a hybrid method of two-stage training, which generates Bash comments through one-stage information retrieval and two-stage CodeBERT fine-tuning.

B. Adversarial Training in NLP

Adversarial training is an important way to enhance the robustness of neural networks. In the process of adversarial training, examples will be mixed with some small perturbations (the change is small, but it is likely to cause misclassification), and then the neural network will adapt to this change, thus being robust to adversarial examples.

The general principle of adversarial training can be summarized as the following maximum-minimum formula:

$$\min_{\theta} \mathbb{E}_{(x,y)} \sim D \left[ \max_{||\delta|| \leq \epsilon} L(f_{\theta}(x + \delta), y) \right]$$ (1)

where $x$ represents input, $\delta$ represents perturbation, $y$ represents the label of the example, and $\max(L)$ represents the optimization objective.

In the NLP (natural language processing) domain, typical adversarial training methods can be categorized as rule-based and gradient-based methods. In terms of rule-based methods, researchers generate adversarial examples by heuristic rules. Ebrahimi et al. [21] proposed a white-box method, which uses character/word substitution to generate adversarial examples. Ribeiro et al. [6] found that adversarial attacks can be useful when debugging NLP models. Cheng et al. [22] also found that model performance can be significantly improved in the field of neural machine translation by generating adversarial examples. In the type of gradient-based methods, researchers generate adversarial examples by adding a perturbation to the Embedding layer. Miyato et al. [9] introduced adversarial training and virtual adversarial training [23] to improve the performance of classification models. Zhu et al. [11] proposed FreeLB for the language model, which promotes higher invariance in the embedding space by adding hostile perturbation to the word embedding and minimizing the risk of result hostility in different regions around the input sample. Recently, Zhou et al. [24] achieved optimal performance using adversarial training and integrated learning techniques by two separately trained encoder-decoder models for source code sequences and corresponding abstract syntax trees (ASTs) in the area of code summarization generation.

C. Research Motivation

Research [5] shows that only 10% of the automatically generated comments can reach the human level. For this reason, we hope to explore a technology that can optimize the model performance without changing code comment generation models, and verify the feasibility of this exploration direction through preliminary results. We consider adversarial training as the experimental object, which can improve the robustness [25] and generalization [26] of the model, and can also be used as a technology for data augmentation [27]. However, to the best of our knowledge, we haven’t found any work related to adversarial training in code comment generation task. To fill this gap, we have done novel experiments to explore the performance of adversarial training in code comment generation.

III. Research Questions

In our empirical study, we aim to answer the following three research questions (RQs).

**RQ1: How efficient is adversarial training on classic deep learning models?**

Motivation. The classical deep learning model is a highly recognized model in earlier studies, and it is often used as a baseline in current studies [16], [28]. We consider them as experimental objects, which can better reflect the rigor of our experiments.

**RQ2: How efficient is adversarial training on pre-trained models?**

Motivation. Code comment generation task is a typical text-to-text problem and PLMs (pre-trained language models) [5], [9], [14] have achieved SOTA in the current task, which is the focus of current researchers. We consider them as experimental objects, which can better reflect the universality of confrontation training.

**RQ3: How efficient is adversarial training on hybrid models?**

Motivation. The hybrid models [5], [15], [16] use deep learning and information retrieval technology to enhance the model representation through retrieval. We consider them as experimental objects, which can better reflect the comprehensiveness of our experimental setup.

IV. Case Study Design

In this section, we describe our experimental setup to address our research questions. We show the overview of our experiment design in Figure 1. This figure mainly shows the process of building code comment generation models through adversarial training method.
A. Experimental Subject

We consider the corpus shared by Yu et al. [16] as the experimental object in our research and continue to do research on more corpus in the future. This high-quality corpus contains 10,592 samples from NL2Bash [29] and NLC2CMD competition. The statistics of corpus are shown in Table I.

### Table I

**LENGTH STATISTICS OF SAMPLES IN THE CORPUS**

<table>
<thead>
<tr>
<th>Code length statistics</th>
<th>Average</th>
<th>Mode</th>
<th>Median</th>
<th>&lt;16</th>
<th>&lt;32</th>
<th>&lt;48</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8.528</td>
<td>4</td>
<td>7</td>
<td>90.8%</td>
<td>99.7%</td>
<td>99.9%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code comment length statistics</th>
<th>Average</th>
<th>Mode</th>
<th>Median</th>
<th>&lt;16</th>
<th>&lt;32</th>
<th>&lt;48</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11.874</td>
<td>10</td>
<td>11</td>
<td>80.3%</td>
<td>99.5%</td>
<td>99.9%</td>
</tr>
</tbody>
</table>

In this table, we find that Bash codes and comments are mostly within 48. In the corpus division, we use the random sampling method [28] to randomly divide the corpus into the training set, evaluation set, and test set according to the ratio of 8:1:1.

B. Performance Measures

In order to quantitatively compare the performance of each code comment generation method, we consider three performance metrics (BLEU [30], METEOR [31] and ROUGE-L [32]) from the research of neural machine translation. These performance metrics are also widely used in the research of automatic code comment generation [33], which can compare the quality of generated comments and reference comments. The higher the calculated performance metric, the better the performance of the corresponding method.

We use the implementation provided by the `nlg-eval` library⁵ to evaluate the performance, which can avoid the difference in results caused by different experiments.

C. Adversarial Training Methods

Adversarial training generates adversarial examples $x_{adv}$ by adding perturbation $\delta$ to model input $x$, where $x_{adv} = x + \delta$. In this experiment, we mainly consider the following four methods of adversarial training that are often used in NLP:

- **FGSM.** FGSM is a method of adversarial training proposed by Goodfellow et al. [8]. The input gradient is $g = \nabla_x L(f_\theta(x), y)$, where $\theta$ is the model parameter value, $x$ is the model input, $y$ is the label, and $L()$ is the loss function of the training model. The perturbation goes to the maximum of the loss function along the gradient direction, which is expressed as $\delta = \epsilon sign(g)$, where $sign()$ is the regularization method, and $\epsilon$ is the constraint that the perturbation is limited by infinite norm (i.e., $\|\delta\|_\infty < \epsilon$).
- **FGM.** FGM is also the method proposed by Goodfellow et al. [8]. Unlike FGSM, which takes the $sign$ function to regularize the gradient, FGM carries out $L2$ regularization on the gradient. In addition, FSGM takes the same step in each direction, while FGM scales according to the specific gradient to get better adversarial examples. The perturbation is expressed as $\delta = \epsilon (g/\|g\|_2)$, where $\epsilon$ is the constraint of perturbation ($L2$ norm of the distance between the original example and the adversarial example is always $\epsilon$).
- **PGD.** FGM calculates the perturbation directly through $epsilon$ parameter, which may not be optimal. Therefore, PGD [10] has been improved and iterated several times to find the optimal perturbation. The input gradient of each step is expressed as $g_t = \nabla_x L(f_{\theta_{t-1}}(x), y)$ and the perturbation of each step is expressed as $\delta_t = \epsilon (g_t/\|g_t\|_2)$. In the iteration, $\delta_t$ is gradually accumulated, and only the gradient calculated by the last $x_t + \delta_t$ is used when the parameters are finally updated.
- **FreeLB.** In order to find the optimal perturbation, Zhu et al. [11] proposed the FreeLB method. PGD takes the gradient $g_t$ of the last perturbation after iterating $K$ times when updating parameters, and FreeLB takes the average gradient $g_{avg}$ of $K$ times of iteration when updating parameters, where $g_{avg} = (g_1 + \cdots + g_t)/t$.

D. Code Comment Generation Methods

To show the feasibility of our proposed research, we have recently considered three types of methods that researchers have focused on (i.e., classical deep learning method, pre-training model method, and hybrid method).

The first type is deep learning-based methods.
- **CODE-NN.** CODE-NN [12] is the first to generate code comments using LSTM and attention mechanism.
- **Transformer.** Transformer [13] is an encoder-decoder model that utilizes the self-attention mechanism.

The second type is pre-trained-based methods.
- **CodeBERT.** CodeBERT [14] builds the model based on the neural architecture of Transformer, and uses the mixed objective function to train the model.
- **UniXcoder.** UniXcoder [14] is a unified cross-modal pre-trained model, which uses a masked attention matrix to control the model and uses cross-modal content to enhance code representation.
- **CodeT5.** CodeT5 [3] supports multi-task learning and can make better use of the information of code tokens to train the model.

The third type is the methods of hybrid information retrieval and deep learning techniques.

---

⁵https://eval.ai/web/challenges/challenge-page/674/leaderboard/1831
⁶https://github.com/Maluuba/nlg-eval
• **Rencos.** Rencos [15] first retrieves similar codes, and then the encoder vectors are fused by the decoder.

• **Hybrid-DeepCom.** Hybrid-DeepCom [8] considers the semantics of Java and traverses AST to obtain syntax information and structure information of the code.

• **BASHEXPLAINER.** BASHEXPLAINER [16] utilizes two-stage training strategies: fine-tuning stage and the information retrieval enhancement stage.

### E. Experimental Settings

We implement our experiment based on Pytorch 2.0. Specifically, we choose AdamW as the optimizer and utilize HuggingFace 4 to implement our selective method. We set the value of epoch to 30 and set early stopping with epoch 5.

We run all the experiments on a computer with an Intel CPU i5-13600K and a GeForce RTX 4090 GPU with 24 GB memory. The running OS platform is Windows OS.

### V. Result Analysis

#### A. RQ1: How efficient is adversarial training on classic deep learning models?

<table>
<thead>
<tr>
<th>Model Name</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE-NN</td>
<td>29.53</td>
<td>24.17</td>
<td>26.85</td>
<td>47.21</td>
</tr>
<tr>
<td>Transformer</td>
<td>25.42</td>
<td>19.97</td>
<td>25.22</td>
<td>44.01</td>
</tr>
</tbody>
</table>

Table II shows the overall results of the different deep learning methods concerning three evaluation measures (i.e. BLEU, ROUGE-L, and METEOR), and we mark the best one of each metric in bold. The first line of the table is without PGD method, and the second line is with PGD method.

According to the experimental results, we find that adversarial training PGD can improve the performance of classic deep learning models. Specifically, compared with not using PGD, the performance of classic deep learning models is improved by 18.89%, 35.35%, 4.56%, and 4.93% for BLEU-3, BLEU-4, METEOR, and ROUGE-L at least. This result indicates that adversarial training, as a special regularization method [35], plays an important role in optimizing the model performance method.

**Answer to RQ1:** Comparison results show that adversarial training can improve the performance of classic deep learning models.

#### B. RQ2: How efficient is adversarial training on pre-trained models?

<table>
<thead>
<tr>
<th>Model Name</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeBERT</td>
<td>29.84</td>
<td>24.83</td>
<td>27.16</td>
<td>47.36</td>
</tr>
<tr>
<td>UniXcoder</td>
<td>31.80</td>
<td>27.25</td>
<td>29.03</td>
<td>48.24</td>
</tr>
<tr>
<td>CodeT5</td>
<td>34.70</td>
<td>29.93</td>
<td>30.29</td>
<td>50.56</td>
</tr>
</tbody>
</table>

Table III shows the overall results of the different pre-trained models concerning three evaluation measures (i.e. BLEU, ROUGE-L, and METEOR), and we mark the best one of each metric in bold. The first line of the table is without PGD method, and the second line is with PGD method.

According to the experimental results, we find that adversarial training PGD can improve the performance of pre-trained models. Specifically, compared with not using PGD, the performance of pre-trained models is improved by 13.37%, 17.92%, 6.59%, and 5.95% for BLEU-3, BLEU-4, METEOR, and ROUGE-L at least. This result indicates that adversarial training, as a data augmentation method [27], greatly improves the pre-trained model that needs to be trained with a large-scale corpus.

**Answer to RQ2:** Comparison results show that adversarial training can improve the performance of pre-trained models.

#### C. RQ3: How efficient is adversarial training on hybrid models?

<table>
<thead>
<tr>
<th>Model Name</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rencos</td>
<td>28.66</td>
<td>24.39</td>
<td>25.82</td>
<td>45.06</td>
</tr>
<tr>
<td>Hybrid-DeepCom</td>
<td>27.91</td>
<td>22.75</td>
<td>26.27</td>
<td>45.36</td>
</tr>
<tr>
<td>BASHEXPLAINER</td>
<td>33.73</td>
<td>29.13</td>
<td>28.78</td>
<td>48.81</td>
</tr>
</tbody>
</table>

Table IV shows the overall results of the different hybrid methods concerning three evaluation measures (i.e. BLEU, ROUGE-L, and METEOR), and we mark the best one of each metric in bold. The first line of the table is without PGD method, and the second line is with PGD method.

According to the experimental results, we find that adversarial training PGD can improve the performance of hybrid models. Specifically, compared with not using PGD, the performance of hybrid models is improved by 9.17%, 14.68%, 4.04%, and 5.37% for BLEU-3, BLEU-4, METEOR, and ROUGE-L at least. The result indicates that the model robustness improved by adversarial training [25] may be helpful to the model that needs multi-stage training.

**Answer to RQ3:** Comparison results show that adversarial training can improve the performance of hybrid models.
VI. DISCUSSION & IMPLICATIONS

In this section, we first study the effect of the adversarial training method considered on code comment generation models. Secondly, we introduce the implication of the experiment and propose a novel adversarial training method based on regularization optimization, and verify the feasibility of this exploration direction through preliminary results.

A. Analysis on Adversarial Training Methods

In this subsection, we want to explore which adversarial training method is suitable for code comment generation. We consider evaluating the performance of different adversarial training methods on CodeBERT. Table V shows the comparison results, and we mark the best one of each metric in bold. The first line of the table is without the adversarial training method, and the other lines are with the adversarial training method.

<table>
<thead>
<tr>
<th>AT Name</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGSM</td>
<td>29.69</td>
<td>25.22</td>
<td>27.12</td>
<td>47.99</td>
</tr>
<tr>
<td>FGM</td>
<td>33.52</td>
<td>29.05</td>
<td>28.10</td>
<td>49.28</td>
</tr>
<tr>
<td>PDG</td>
<td>34.55</td>
<td>29.98</td>
<td>29.31</td>
<td>49.86</td>
</tr>
<tr>
<td>FreeLB</td>
<td>33.83</td>
<td>29.28</td>
<td>28.95</td>
<td>49.67</td>
</tr>
</tbody>
</table>

According to the experimental results, we find that the performance of CodeBERT can be improved to the maximum by using PGD method. Specifically, compared with the method with the lowest index, CodeBERT can improve the performance by 16.36%, 18.87%, 8.08%, and 3.90% for BLEU-3, BLEU-4, METEOR, and ROUGE-L respectively. This result indicates that only the last iteration of PGD method is more suitable for the code comment generation task.

B. Practical Guidelines

Our experimental results show that adversarial training technology can affect the performance of the model, especially for the complex pre-trained model. In this subsection, we provide practical guidelines for future research on code comment generation.

Researchers can consider improving the performance of code comment generation models from outside the model. The results in Section 3 show that adversarial training technology can significantly improve the performance of code comment generation models. As a special regularization method [55], adversarial training can optimize model parameters by generating adversarial examples without changing the model. We suggest that future researchers can optimize code comment generation models by using adversarial training technology.

Researchers can change the regularization method to make adversarial training more suitable for the code comment generation task. As mentioned in the previous paragraph, as a special regularization method [55], the optimization loss function can be expressed as \( \bar{L}(x, y) = L(x, y) + \frac{\epsilon}{2} \| \partial_x L \|_p \), where \( x \) represents the input, \( y \) represents the label, \( \bar{L} \) represents the loss after joining the adversarial training, \( L \) represents the original loss, and \( \frac{\epsilon}{2} \| \partial_x L \|_p \) represents the special regularization term. This shows that the model performance can be further optimized by changing the regularization method. On this basis, we propose a novel adversarial training method, normPGD, which is oriented to the code comment generation task. Limited by space, the related introduction and results are on the project homepage 3. We suggest that future researchers can customize the adversarial training method which is more suitable for the code comment generation task by changing the regularization term.

VII. THREATS TO VALIDITY

Internal threats. The main first internal threat is the implementation correctness of code comment generation models we chose. To alleviate this threat, we re-implement their method according to the description of their empirical research and achieve similar performance. The second internal threat is the potential defect of the experimental model we designed. To alleviate this threat, we use mature libraries to implement code, such as Pytorch and Transformers.

External threats. The threat of external validity is the choice of experimental subjects. In order to alleviate this threat, we chose the corpus provided by Yu et al. [15]. Yu et al. improved the quality of data pairs by pre-processing the corpus. At the same time, this corpus has also been used in the research of code comment generation. In the future, we hope to verify the performance of our questions in popular languages (such as Java, Python, etc.).

Construct threats. The construct threat in this study is the performance measures used to evaluate our proposed experiments performance. In order to alleviate this threat, we choose three popular performance metrics (i.e. BLEU [30], METEOR [31], and ROUGE-L [32]) in the field of neural machine translation. These evaluation metrics are also widely used in the field of code comment generation [53].

Conclusion threats. The conclusion threat in our study is mainly that there is no cross-validation. Due to the high training cost of deep learning, this method has not been widely used in the field of neural machine translation. In this study, we only divide the corpus once, which is consistent with some previous studies on code comment generation [15].

VIII. CONCLUSION

This paper gives a comprehensive evaluation of the adversarial training methods in the field of code comment generation. Specifically, we first collect the existing popular adversarial training methods and find out the most suitable method for the code comment generation task through experiments. Then, we propose three types of research questions and conduct experiments on them. Finally, we draw a conclusion about the performance of adversarial training in the code comment generation task. We believe that this study can help to further evaluate and improve the code comment generation task. The

3https://github.com/syhstudy/AT_Empirical_Study
limitation of this work lies in the limited corpus, models, and metrics. This can draw the help of adversarial training as a data augmentation method better for low-resource tasks. Future work may be carried out by collecting more corpus or evaluating more popular (such as Java, Python, etc.) corpus to more accurately evaluate the performance of adversarial training in the code comment generation task.

ACKNOWLEDGEMENTS

This work is supported in part by the Jiangsu Province Modern Educational Technology Research Project under Grant No. 2022-R-98984 and the Nantong Application Research Plan under Grant No. J221087.

REFERENCES

No-Reference Point Cloud Quality Assessment via Weighted Patch Quality Prediction

Jun Cheng  
Shenzhen University  
chengjun2020@email.szu.edu.cn  
Honglei Su*  
Qingdao University  
suhonglei@qdu.edu.cn  
Jari Korhonen  
University of Aberdeen  
jari.t.korhonen@ieee.org

Abstract—With the rapid development of 3D vision applications based on point clouds, point cloud quality assessment (PCQA) is becoming an important research topic. However, the prior PCQA methods ignore the effect of local quality variance across different areas of the point cloud. To take an advantage of the quality distribution imbalance, we propose a no-reference point cloud quality assessment (NR-PCQA) method with local area correlation analysis capability, denoted as COPP-Net. More specifically, we split a point cloud into patches, generate texture and structure features for each patch, and fuse them into patch features to predict patch quality. Then, we gather the features of all the patches of a point cloud for correlation analysis, to obtain the correlation weights. Finally, the predicted qualities and correlation weights for all the patches are used to derive the final quality score. Experimental results show that our method outperforms the state-of-the-art benchmark NR-PCQA methods. The source code for the proposed COPP-Net can be found at https://github.com/philox12358/COPP-Net.

Index Terms—Point convolution, point cloud quality assessment, deep learning

I. INTRODUCTION

A 3D point cloud is a large and dense collection of sampled points with spatial coordinates and attributes, obtained by 3D scanning technology like lidar. Each point in a point cloud contains a geometric attribute, i.e. 3D space coordinates, and other attributes, such as color, reflectivity, opacity, etc., represented by feature vectors. As point clouds directly represent the 3D world, they are widely used in automatic driving, industrial robots, cultural heritage protection, geographic mapping, and other fields.

Similar to image and video, point clouds can be distorted due to many factors during the collection and transmission process. However, point clouds suffer from more complex distortions due to their data format. Therefore, PCQA is more challenging than traditional image quality assessment. Accurate PCQA is critical for providing high-quality point clouds for various purposes. Like in traditional image and video quality assessment, PCQA methods can also be divided in full-reference (FR), reduced-reference (RR), and no-reference (NR) methods, according to the availability of a reference point cloud.

Several FR-PCQA methods have been proposed recently [1] [2] [3] [4] [5] [6], and they have reached relatively good performance already. However, in many real-life application scenarios, there is no pristine reference point cloud available, and NR-PCQA is therefore a very important research topic at the moment. Regarding the recent development of image quality assessment (IQA), the most accurate NR-IQA methods are based on deep convolutional neural networks (CNN). Unfortunately, training of deep neural networks (DNN) requires a large amount of training samples, and the currently available PCQA databases are mostly small, and each point cloud often contains millions of points. Advanced point convolution neural networks directly processing such a large number of points have a very high computational load, limiting the use of deep neural networks for NR-PCQA.

Although some attempts have been made to use DNNs for NR-PCQA [8] [9] [10], these methods usually obtain projected 2D images from the whole point cloud or a few features of the whole point cloud as the input of the network through preprocessing, rather than directly using the point cloud. ResSCNN [11] directly uses the entire point cloud as input, but sparse convolution is used to save computation, which reduces the computational accuracy. NR-CNN-3D-PC [12] divides the point cloud into many patches, but does not consider the impact of different patches on the overall quality of the point cloud. At the same time, compared with the most accurate FR-PCQA method, the proposed models still perform poorly. So far, we have not identified any stable and accurate method...
of NR-PCQA task that uses point cloud as the input of DNN model directly and takes into account the local correlation of point cloud.

Due to the development of new methods for deep learning in the recent years, such as PointNet++ [13] and transformers [14], there have been remarkable progress in processing point clouds with fewer points recently. On the other hand, in many practical use cases large 3D point clouds need to be observed from different sides and viewing angles multiple times through rotation and scaling. Moreover, point cloud usually has complex structure and rich texture, and the visual appearance of different regions can be very different. Therefore, the correlation between the local quality and the overall quality varies strongly between different areas of the point cloud. Based on these considerations and the previous work, we propose a patch-based NR-PCQA method that generates features representing the structure and texture of each patch. Then, point cloud quality is predicted by using appropriate weights derived from the correlations between patch quality and overall quality.

It is worth noting that correlation in our method is different from saliency. Even though the correlations may be related to saliency, our method does not aim to calculate the saliency map of the point cloud. Specifically, we first divide a point cloud into several patches, representing different parts of the point cloud. In the proposed COPP-Net model, we propose Adaptive R-Sampling KNN PointNet++ (ARKP) network, based on PointNet++ architecture [13], to generate texture and structure features from patches generated from the point cloud. These features are used to predict local quality scores for the patches, and they are also used as input to the proposed Quality Correlation Analysis (CORA) network, predicting the weights for each patch. Finally, the quality scores for each patch are multiplied by their relative weights to predict the overall quality score through a weighted average.

II. RELATED WORK

A. Full-Reference metrics

For PCQA, several FR metrics were firstly developed to be used for assessing performance of point cloud compression. The classic methods are point-based FR metrics proposed by MPEG, such as p2point [1] and p2plane [15]. Their computational complexity is low, but on the other hand, their accuracy is limited and they are prone to instability when complex distortion types are concerned. In [16], a point-based metric based on angular similarity was proposed, and [17] proposed a series of methods using local luminance patterns, local binary patterns and multi-distance approach. In [18], a generalized Hausdorff distance based quality metric for geometric point cloud distortions was proposed, and [17] used the local curvature statistics to evaluate the quality of point clouds. Later, inspired by SSIM [4], several metrics were proposed in [19][20]. They considered the structural characteristics of the point cloud and achieved a good performance for FR-PCQA.

B. No-Reference metrics

Due to the limited availability of point cloud quality databases, exploration of NR-PCQA started relatively late. Since NR-IQA methods have matured [8], it has been proposed to project point clouds to 2D pictures from different angles, and then use traditional IQA methods or CNN to predict point cloud quality indirectly from the 2D images. In NR-3DQA [9], 3D point clouds are projected into feature domains based on color and geometry, and the quality score is obtained using support vector machine (SVM) regression. IT-PCQA [10] using the rich prior knowledge of natural images to build a bridge between 2D and 3D perception for quality assessment via transfer learning. ResSCNN [11] uses 3D sparse convolution...
for efficient computations. However, in the process of sparse convolution, the local accuracy of the point cloud data will be reduced, resulting in partial information loss. Therefore, the performance of the method is not stable on point clouds with different types of distortions.

In [12], point clouds are split into multiple local patches, and low level patch-wise features (e.g., geometric distance, local curvature) are used to train a CNN. However, this method does not take into account the quality inconsistency between different areas of the point cloud. To tackle this challenge, we propose a model that learns the weights for different parts of the point cloud to balance the inconsistencies in local quality.

III. OUR APPROACH

A. Point Cloud Preprocessing Module

A point cloud instance is a set \( P \) that contains a total of \( N \) points, \( P = \{ P_i \mid i = 1, 2, ..., N \} \), located on the surface of the object of interest. Note that \( P_i = (x_i, y_i, z_i, r_i, g_i, b_i) \) defines the \( i \)-th point of this point cloud, and the spatial coordinates denoted as \( (x, y, z) \) and the color information denoted as \( (r, g, b) \). Thus, a 3D point cloud object can be represented by a matrix of size \( N \times 6 \). Generally, \( N \) is in millions. Processing that many points will consume a lot of computing resources. Considering that different parts of the point cloud have different quality correlations, we divide the point cloud into multiple patches through preprocessing.

For preprocessing, we first normalize \((x, y, z)\) into a sphere with a radius of 1000, then we use the farthest point sampling (FPS) [13] algorithm to obtain \( C \) center points, and finally we use the \( k \)-nearest neighbor (KNN) [19] algorithm to sample the nearest \( K \) points to each center point to form the patches. For each patch, we center the \((x, y, z)\) coordinates and the \((r, g, b)\) color information, respectively. As a result, we sample \( C \) patches from a point cloud, and each patch contains \( K \) points. These patches are then used as input to the patch feature generation module.

B. Patch Feature Generation Module

The feature generation module will generate features for texture and structure information. Texture information is very sensitive to downsampling, because it requires fine and continuous dense data representation. Fortunately, continuous regions with similar quality scores tend to share similar texture distortion characteristics. Therefore, we can generate texture features from small regions without losing the accuracy of the representation. For structural information, rough and sparse information is sufficient to represent complex spatial structures, so structural information is not sensitive to downsampling. Based on these observations, we follow different strategies to generate texture and structure features.

Specifically, the module is composed of local texture feature generation network (ARKP\(_t\)) and 3D structure feature generation network (ARKP\(_s\)). Both networks are based on ARKP architecture, and they are used to generate texture and structure features, respectively. The details are as follows:

- **Texture feature generator ARKP\(_t\)**. In a patch with \( K \) points, we use KNN algorithm to sample locally adjacent and continuous \( R_t \) points to be used as input to ARKP\(_t\) for obtaining the texture features of the patch.

  Then, we concatenate the features from the two networks and send it through a MaxPool layer to obtain the final patch feature vectors \( F_{patch} \). \( F_{patch} \) will be used to predict the patch quality score \( Q_{patch} \). It is also used by the CORA network to obtain correlation weight \( W_{patch} \). We will present a detailed description of the ARKP network below.

- **Structure feature generator ARKP\(_s\)**. In a patch with \( K \) points, we use random sampling to sample a relatively small number of \( R_s \) points within the range of the patch to be used as input to ARKP\(_s\) for obtaining the structure features of the patch.

  The ARKP network differs from the SSG version of PointNet++ in the following aspects:

![AR KP Network](image_url)

Fig. 3. Detailed structure of ARKP network.

![Overall point cloud MOS (red) and predicted Q\(_{patch}\) for patches (blue) on WPC database.](image_url)

Fig. 4. Overall point cloud MOS (red) and predicted \( Q_{patch} \) for patches (blue) on WPC database.
Our ablation experiments demonstrate the effectiveness of the proposed modifications.

D. Point Cloud Quality Regression Module

Figure 2 shows a detailed overview of the steps to predict the point cloud quality score from the \( F_{\text{patch}} \). Using \( F_{\text{patch}} \) as input, we apply a regression head comprising two linear layers, batch normalization layer, and leaky ReLU layer to predict \( Q_{\text{patch}} \). In the training phase, we assign the overall point cloud quality score as ground truth quality score for all the patches. Mean squared error (MSE) is used as loss function for training. If the quality score of a point cloud is calculated by averaging the values of \( Q_{\text{patch}} \), high prediction accuracy seems to be achieved.

However, as shown in Figure 4, the quality scores for individual patches tend to be scattered, since different areas of the point cloud have different quality levels. To reduce the impact of outlier patches, we use a network (CORA) to analyze the dispersion of quality of each patch.

E. CORA Network

The CORA network is designed to estimate the correlation between the patch quality and the overall point cloud quality by predicting correlation labels. As illustrated in Figure 5, we concatenate all \( F_{\text{patch}} \) of a single point cloud to form the input to the CORA network. Then, we apply a multilayer perceptron (MLP) with two linear layers, followed by four transformer blocks and two additional linear layers to predict the correlations. The middle hidden layer dimension is set to 512. Finally, SoftMax operation is applied to the output to compute the correlation-based \( W_{\text{patch}} \).

Specifically, we obtain all \( Q_{\text{patches}} \) of a point cloud and rank them in ascending order based on the absolute difference between each \( Q_{\text{patch}} \) and the ground truth quality score of the point cloud. Then divide them into three correlation classification labels, i.e., strong correlation, average correlation, and weak correlation. The goal of the CORA network is to predict the correlation classifications accurately. We use cross-entropy loss to train the CORA network.

In the final phase of computing the predicted overall quality score, we use correlation weight pooling. Specifically, we use \( W_{\text{patches}} \) from the CORA network, and then we introduce the Waterloo Point Cloud Database (WPC), and compare the performance of our COPP-Net with the prior mainstream FR and NR methods. Then, we conduct an ablation study on the proposed COPP-Net. Finally, we test the performance of the model on other databases.

IV. EXPERIMENT

In this section, we first describe the experimental setup, then we introduce the Waterloo Point Cloud Database (WPC), and compare the performance of our COPP-Net with the prior mainstream FR and NR methods. Then, we conduct an ablation study on the proposed COPP-Net. Finally, we test the performance of the model on other databases.

A. Experiment Setups

In each of the practical experiments, we extracted \( C=16 \) patches from a point cloud, with \( K=14900 \) points in each patch. In the feature generation module, we used KNN algorithm to select \( R_t=8192 \) adjacent points around the center of a patch as input to ARKP, and random sampling to select \( R_s=1024 \) points as input to ARKP.

The model was trained in two stages: in the first stage, we trained the feature generation and patch quality prediction part, and in the second stage, we trained the CORA network.

We used 32 mini-batches in the first stage and 4 in the second stage. In both training stages, we used stochastic gradient descent (SGD) optimization algorithm. We set the initial learning rate to \( 10^{-4} \) and applied cosine learning rate decay method until the loss converged. As performance indicators for the final model, we used Pearson correlation coefficient (PLCC), Spearman rank order correlation coefficient (SRCC) and root mean squared error (RMSE) to evaluate the accuracy of the predicted scores in comparison to the ground truth quality scores. These indicators are all widely used to assess the

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>PLCC</th>
<th>SRCC</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR</td>
<td>NIQE [20]</td>
<td>0.3957</td>
<td>0.3887</td>
<td>22.55</td>
</tr>
<tr>
<td></td>
<td>BRISQUE [21]</td>
<td>0.4176</td>
<td>0.3781</td>
<td>22.54</td>
</tr>
<tr>
<td></td>
<td>PQA-Net [8]</td>
<td>0.7000</td>
<td>0.6900</td>
<td>15.18</td>
</tr>
<tr>
<td></td>
<td>NR-3DQA [9]</td>
<td>0.6591</td>
<td>0.6526</td>
<td>16.45</td>
</tr>
<tr>
<td></td>
<td>ResSCNN [11]</td>
<td>0.4292</td>
<td>0.4352</td>
<td>23.27</td>
</tr>
<tr>
<td></td>
<td>IT-PCQA [10]</td>
<td>0.5609</td>
<td>0.5683</td>
<td>19.07</td>
</tr>
<tr>
<td></td>
<td>Our COPP-Net</td>
<td>0.9324</td>
<td>0.9251</td>
<td>8.10</td>
</tr>
</tbody>
</table>
TABLE II
COMPARISON OF OUR COPP-Net WITH THE FR METHOD ON THE TEST SOURCE CONTENTS IN WPC DATABASE. IN ALL CASES, THE PROPOSED METHOD PERFORMS BETTER THAN THE FR METHODS. THE RESULTS FOR THE FR METHODS ARE FROM [7].

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Banana</th>
<th>Cauliflower</th>
<th>Mushroom</th>
<th>Pineapple</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PLCC</td>
<td>SRCC</td>
<td>RMSE</td>
<td>PLCC</td>
</tr>
<tr>
<td></td>
<td>PSNRt w/o R-Sampling M [1]</td>
<td>0.7236</td>
<td>0.8471</td>
<td>14.98</td>
<td>0.4555</td>
</tr>
<tr>
<td></td>
<td>PSNRt [1]</td>
<td>0.7560</td>
<td>0.6785</td>
<td>14.20</td>
<td>0.6338</td>
</tr>
<tr>
<td></td>
<td>PSNRt w/o R-Sampling [1]</td>
<td>-0.7145</td>
<td>-0.7686</td>
<td>15.18</td>
<td>-0.7057</td>
</tr>
<tr>
<td></td>
<td>FF [1]</td>
<td>0.7156</td>
<td>0.7544</td>
<td>15.16</td>
<td>0.6515</td>
</tr>
<tr>
<td></td>
<td>GraphSIM [5]</td>
<td>0.5990</td>
<td>0.5300</td>
<td>17.38</td>
<td>0.6010</td>
</tr>
<tr>
<td></td>
<td>VIFP [6]</td>
<td>0.7938</td>
<td>0.7956</td>
<td>13.20</td>
<td>0.6581</td>
</tr>
<tr>
<td></td>
<td>IW-SSIM [7]</td>
<td>0.8724</td>
<td>0.8627</td>
<td>10.61</td>
<td>0.8578</td>
</tr>
<tr>
<td></td>
<td>PQA-Net [8]</td>
<td>0.8589</td>
<td>0.8446</td>
<td>10.42</td>
<td>0.9324</td>
</tr>
<tr>
<td></td>
<td>NR-3DQA [9]</td>
<td>0.6965</td>
<td>0.7008</td>
<td>16.13</td>
<td>0.5977</td>
</tr>
<tr>
<td></td>
<td>ResSCNN [11]</td>
<td>0.4484</td>
<td>0.4694</td>
<td>25.04</td>
<td>0.4113</td>
</tr>
<tr>
<td></td>
<td>IT-PCQA [10]</td>
<td>0.7855</td>
<td>0.8377</td>
<td>13.58</td>
<td>0.7156</td>
</tr>
<tr>
<td></td>
<td>Our COPP-Net</td>
<td>0.9121</td>
<td>0.8878</td>
<td>9.07</td>
<td>0.9190</td>
</tr>
</tbody>
</table>

TABLE III
ABLATION STUDY OF THE COPP-Net MODEL ON THE WPC DATABASE. AVE STANDS FOR AVERAGE POOLING, AND CORA STANDS FOR CORRELATION WEIGHT POOLING STRATEGY.

<table>
<thead>
<tr>
<th>Model</th>
<th>Pooling</th>
<th>Criteria</th>
<th>PLCC</th>
<th>SRCC</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet++ SSG [13]</td>
<td>AVE</td>
<td></td>
<td>0.7136</td>
<td>0.6988</td>
<td>15.65</td>
</tr>
<tr>
<td>ARKP t w/o Stride Conv</td>
<td>AVE</td>
<td></td>
<td>0.7724</td>
<td>0.7653</td>
<td>14.86</td>
</tr>
<tr>
<td>ARKP t w/o R-Sampling</td>
<td>AVE</td>
<td></td>
<td>0.8461</td>
<td>0.8286</td>
<td>13.74</td>
</tr>
<tr>
<td>ARKP t w/o KNN</td>
<td>AVE</td>
<td></td>
<td>0.7538</td>
<td>0.7346</td>
<td>15.03</td>
</tr>
<tr>
<td>ARKP</td>
<td>AVE</td>
<td></td>
<td>0.8589</td>
<td>0.8446</td>
<td>11.75</td>
</tr>
<tr>
<td>ARKP s</td>
<td>AVE</td>
<td></td>
<td>0.8691</td>
<td>0.8616</td>
<td>10.91</td>
</tr>
<tr>
<td>ARKP t+ARKP s</td>
<td>AVE</td>
<td></td>
<td>0.9015</td>
<td>0.8994</td>
<td>9.92</td>
</tr>
<tr>
<td>ARKP t+ARKP s+CORAM</td>
<td>CORA</td>
<td></td>
<td>0.9324</td>
<td>0.9251</td>
<td>8.10</td>
</tr>
</tbody>
</table>

performance of IQA and PCQA models. All our experiments were run on a single NVIDIA RTX 3090 GPU.

B. Prediction performance on WPC database

Waterloo Point Cloud Database (WPC) includes 20 high-quality color point clouds, covering various geometric and texture characteristics. The database includes 37 distorted versions for each original point cloud, and there are in total 20 original point clouds and 740 distorted point clouds in the database. Each point cloud is annotated with their ground truth quality scores, obtained by subjective testing. In our experiments, we used the same division into training and testing sets as described in [8] [9]: the point clouds for Banana, Cauliflower, Mushroom, Pineapple, accounting for 20% of the whole data set, were used as test set, and the remaining point clouds were used as training set.

Table I lists the overall results of COPP-Net and the selected benchmark NR methods [8] [9] [11] [10] on the WPC test set. The results show that the proposed COPP-Net outperforms the prior state-of-the-art NR methods with a large margin.

In Table II, we summarize the performance of several FR methods [1] [2] [3] [4] [5] [6] [7] on different test contents in the WPC database separately, as given in [7], as well as the results for the same contents with the state-of-the-art NR models [8] [9] [11] [10], and the proposed COPP-Net obtained from our experiments. According to the results, COPP-Net is not only superior to the best performing prior NR methods in each category, but also outperforms the strongest FR method on all source contents. In general, FR models perform better than NR models, because they use the original pristine point cloud as a reference.

C. Ablation Study

To validate the efficacy of the proposed ARKP and CORA networks, we performed experiments on the WPC database using COPP-Net with various network configurations and pooling strategies. The results presented in Table III demonstrate the effectiveness of the modifications we introduced in the ARKP networks. Specifically, the ARKP and ARKP s models exhibit superior feature generation capabilities in comparison to the baseline PointNet++ SSG. Our analysis reveals that the Adaptive Stride Convolution and KNN components play a significant role in boosting performance, whereas the R-sampling component has a minor impact on accuracy, but it contributes to increasing the training speed.

Table III demonstrates that the combination of ARKP t and ARKP s outperforms the individual networks. This indicates that the features generated by these two networks capture different aspects of quality, highlighting the importance of using distinct features for texture and structure. Furthermore, the use of weighted average pooling instead of average pooling in the CORA network enhances the performance, indicating that the model successfully leverages the quality imbalances across various regions of the point cloud.

D. Prediction performance on other databases

To provide a more comprehensive evaluation of the proposed model, we conducted comparative studies using COPP-Net and three state-of-the-art NR methods on five additional publicly available point cloud quality databases, including SJTU-PCQA [22], SIAT-PCQD [23], WPC2.0 [24], and LS-PCQA parts I [11]. In each dataset, we adopted an 80%-20% split for training and testing, respectively. PQA-Net [8] is not included, because it cannot be used on databases with multiple distortion types.

As shown in Table IV, our COPP-Net outperforms most of the other methods across multiple databases, and even in cases where it does not achieve the best performance, the difference
against the best performing method is typically small. These results highlight that the proposed method is widely applicable on different databases.

V. CONCLUSIONS

In this paper, we proposed a novel method for NR point cloud quality assessment, named as COPP-Net. The method takes into account the impact of different quality levels in different parts of the point cloud to the overall quality by using CORA network to constrain the dispersion of local quality levels. The ARKP network used in COPP-Net shows stronger feature generation capability than the baseline, demonstrating the importance of generating both texture and structure features of the point cloud. The experimental results show that the proposed method outperforms state-of-the-art FR and NR methods for PCQA.

The main limitation of COPP-Net is its limited scalability when increasing the number of patches: the CORA network needs to calculate all $F_{\text{patch}}$ features of a point cloud in parallel, which requires a significant amount of GPU memory. In contrast, the proposed ARKP network requires much less GPU memory, and the ARKP network without CORA can also achieve satisfactory performance.

### REFERENCES


### TABLE IV

<table>
<thead>
<tr>
<th>Method</th>
<th>SJTU-PCQA</th>
<th>SLAT-PCQD</th>
<th>WPC-20</th>
<th>LS-PCQA Part I</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PLCC</td>
<td>SRCC</td>
<td>RMSE</td>
<td>PLCC</td>
</tr>
<tr>
<td>NR-3DQA</td>
<td>0.5987</td>
<td>0.5676</td>
<td>2.15</td>
<td>0.0227</td>
</tr>
<tr>
<td>ResSCNN</td>
<td>0.9261</td>
<td>0.9099</td>
<td>1.49</td>
<td>0.3083</td>
</tr>
<tr>
<td>IT-PCQA</td>
<td>0.5922</td>
<td>0.5269</td>
<td>2.09</td>
<td>0.4266</td>
</tr>
<tr>
<td>Our COPP-Net</td>
<td>0.9257</td>
<td>0.8915</td>
<td>1.31</td>
<td>0.8512</td>
</tr>
</tbody>
</table>


GraphPLBART: Code Summarization Based on Graph Embedding and Pre-Trained Model

Jie Li
School of Computer Science and Technology
Soochow University
Jiangsu, China
2027407061@stu.suda.edu.cn
Hao Zhu
School of Computer Science and Technology
Soochow University
Jiangsu, China
2027406032@stu.suda.edu.cn
Lixuan Li
School of Computer Science and Technology
Soochow University
Jiangsu, China
li_xian@outlook.com
Xiaofang Zhang
B
School of Computer Science and Technology
Soochow University
Jiangsu, China
xfzhang@suda.edu.cn

Abstract—Code summarization is a task that aims at automatically producing descriptions of source code. Recently many deep-learning-based approaches have been proposed to generate accurate code summaries, among which pre-trained models for programming languages have achieved promising results. It is well-known that source code written in programming languages is highly structured and unambiguous. Though previous work pre-trained the model with well-designed tasks to learn universal representation from a large scale of data, they haven’t considered structure information during the fine-tuning stage. To make full use of both the pre-trained programming language model and the structure information of source code, we utilize Flow-Augmented Abstract Syntax Tree (FA-AST) of source code for structure information and propose GraphPLBART – Graph-augmented Programming Language and Bi-directional Auto-Regressive Transformer, which can effectively introduce structure information to a well-pre-trained model through a cross attention layer. Experimental results show that our approach outperforms the baseline models in some metrics.

Index Terms—code summarization, pre-trained model, code structure, deep learning

I. INTRODUCTION

Code summary refers to a natural language description of a code segment which can facilitate code comprehension. However, writing high-quality code summaries is a very time-consuming activity, and due to the rapid update of software, many human-written code summaries can be outdated fast. Thus with the development of software engineering, the code summarization task, which aims at automatically generating natural language descriptions for code segments, has received increasing interest in recent years.

Many approaches have been proposed to generate code summaries automatically, among which the deep-learning-based approaches have been proven to be the most effective. For example, Iyer et al. [1] first utilized Long Short Term Memory (LSTM) networks with attention mechanism to produce comments for C# and SQL, which outperformed the traditional retrieval-based method. Ahmad et al. also used a transformer model [2] with copy mechanism [3] to capture the long-range dependencies in source code [4], which proved the transformer model better on the task.

However, in contrast to natural language, source codes are unambiguous and structured [5]. From one aspect, programming languages are formal languages so the structure information is essentially important for deep learning models to learn the representation of source code better. To take advantage of rich and unambiguous structure information of source code, Hu et al. [5] proposed a special structure-based traversal technique to linearize the Abstract Syntax Tree (AST), the linearized ASTs are then fed into an LSTM model to generate summaries. From another aspect, though Transformer has been proven better on code summarization task, it needs more training data as its multi-head attention is purely data-driven [6]. SG-Trans [7] and SiT [8] utilize structure information such as AST and Data Flow Diagram (DFD) as inductive bias to overcome attention collapse or attention redundancy in the Transformer model, which is the main issue that hinders the transformer model’s representation ability [9]. Another solution to the attention collapse problem is pre-training. For example, CodeBERT is a Roberta-based pre-trained model for programming languages and natural language (PL-NL) [10]. The model is pre-trained with the task of masked language modeling [11] and replaced token detection [12] on a large scale of training data. Pre-trained PL-NL model can be easily fine-tuned on a limited dataset for downstream tasks because the pre-training procedure enables the model to capture important dependencies from a large scale of data and thus overcome the attention collapse problem.

Though pre-trained models learn representations from a large scale of training data and have achieved great success on code summarization tasks, they haven’t considered structure information during the fine-tuning stage. To address this limitation and take full advantage of both code structure information and pre-trained models, in this paper, we propose
GraphPLBART: a graph-augmented code summarization approach based on the PLBART pre-trained model [13]. We first parse source codes into ASTs and then construct Flow-Augmented Abstract Syntax Trees (FA-ASTs) [14] by adding additional semantic edges to each AST. Then we use Gated Graph Neural Network (GG-NN) [15] to learn the structure information of source codes. The structure representations are fused into the model with an additional cross-attention layer added after every self-attention layer of the PLBART encoder. The main contributions of this article are as follows:

- A novel graph-augmented code summarization approach is proposed to explore the integration of structure information of source code and PL-NL pre-trained model during the fine-tuning stage.
- Extensive experiments are conducted to prove the effectiveness of introducing structure information during the fine-tuning stage. The experiment results show that our GraphPLBART outperforms the baseline models.

The rest of this paper is organized as follows. Section II introduces the background and our motivation. Our approach is introduced in Section III. The experimental setup and results are presented in Section IV and V respectively. Section VI is the conclusion and the future work.

II. BACKGROUND

A. Pre-trained Programming Language Model

Pre-trained programming language models, which are trained on large scale of data in a self-supervised manner to learn universal programming language representations, can avoid training a new model from scratch and have shown advantages on various downstream tasks. For example, CodeT5 [16] makes use of both PL-only and NL-PL bimodal data to pre-train the model, which yields better results on five downstream tasks. Other pre-trained programming language models e.g., CodeBERT [10] and GraphCodeBERT [17] also demonstrate their promise on program understanding and generation. Besides, PLBART [13] is a sequence-to-sequence model capable of performing various programming and language understanding and generation tasks. It has undergone pre-training of denoising autoencoders on a large number of Java and Python functions, as well as related natural language texts.

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 [18]. Every node of the tree represents a structure of source code. Developers can get the declaration statements, assignment statements, operation statements and realize operations by analyzing the tree structures [19]. Nowadays, AST has been widely used for various program understanding and generation tasks [7] [8] since it contains unambiguous structure information of source code which is essentially important to program understanding.

C. Motivation

Pre-trained language models for programming have shown impressive results in several programming understanding tasks, including code summarization. GraphCodeBERT has implemented an edge prediction pre-training task to learn representations from Data Flow Diagrams, which contain semantic structural information about source code. Although these models learn universal representations from well-designed pre-training tasks on large-scale data, it is essential to include structural information as an inductive bias during the fine-tuning stage. SG-Trans [7] and SiT [8] have attempted to introduce structural information into their approach, but their models are trained from scratch, which can be very time-consuming due to the purely data-driven nature of multi-head attention [6].

To address these limitations, we propose GraphPLBART, a model that effectively incorporates structure information extracted from FA-AST into pre-trained PLBART. During the fine-tuning stage, we add an additional cross-attention layer to our model to incorporate the structural information. This approach allows us to leverage the benefits of pre-training while also accounting for the structural properties of the input code. By doing so, we expect to achieve better performance in programming understanding tasks such as code summarization.

III. APPROACH

GraphPLBART contains a graph reader, an encoder, and a decoder. The graph reader reads grammatical information from the FA-AST and passes it on to the encoder via the attention mechanism. In response, we add another cross-attention layer to the encoder after the original self-attention layer to receive grammatical information. Besides, the weights of the encoder embedding layer, as well as the decoder, are derived from the pre-trained PLBART. The framework of the model is shown in Figure 1.

A. Graph Reader

We first parse the code into an abstract syntax tree (AST) and add data flow and control flow to extend AST into FA-AST (Flow-Augmented Abstract Syntax Tree). Then GraphPLBART extracts information using GG-NN. As with the Transformer, we perform a non-linear transformation of the graph reader with a Feed Forward layer at the end of the graph reader.

1) Flow-Augmented Abstract Syntax Tree: We build the graph representation for programs as follows. To parse ASTs from Java programs, we use a python package javalang. On the other hand, we use a built-in Python package ast to parse from Python programs. It is important to note that although both tools can extract ASTs, they differ in naming the nodes. Therefore, we need to find the corresponding relationship between these two naming methods, such as WhileStatement.

1 https://github.com/2nes/javalang
2 https://github.com/att/ast
Fig. 1: Details of Model. We use a graph reader to extract grammar information from flow-augment AST, then deliver the grammatical message to the encoder by attention mechanism.

in javalang, which becomes While in ast, and there is no direct corresponding naming for BlockStatement in ast. During the code augmentation process, in terms of data flow, we connect non-leaf nodes with their sibling nodes, nodes with their next usage, and leaf nodes with their next nodes. In terms of control flow, we connect the body and condition of *while* and *for* statements, and for *if* statements, we connect the condition to both the true and false branches. Following [14], we add an additional backward edge for each edge that does not have one to increase the frequency of message passing. Figure 2 is an example of extracting AST from Java code and transforming it into FA-AST. The code provided is used to calculate the absolute value of an integer, including function definition, variable calculation, and *if* conditional statement. The black-directed edges in the figure are edges in the AST, and the other colored edges are edges generated during the flow augmentation process. The meaning of each edge is shown in the legend.

2) Gated Graph Neural Network: We use GG-NN only to learn embeddings of each node in FA-AST, so we do not need a readout function to read the information of the entire graph. The calculation formulas of GG-NN are as follows:

$$ m_{j \rightarrow i} = \text{MLP}(h^{(t)}_i, h^{(t)}_j, e_{j,i}), \forall (j, i) \in E $$

$$ m_i = \sum_j m_{j \rightarrow i} $$

$$ h^{(t+1)}_i = \text{GRU}(m_i, h^{(t)}_i) $$

where $m_{j \rightarrow i}$ represents passing message from node $j$ to node $i$, $e_{j,i}$ represents weight of the edge connecting node $j$ and node $i$, $E$ is the set of edges, $h^{(t)}_i$ represents hidden state of node $i$ at time $t$, and $m_i$ represents message of node $i$. MLP and GRU represents multilayer perceptron [20] and gated recurrent unit [21] respectively. To keep more of the original information in the embeddings, we only use one layer of graph network to avoid interference from the embeddings of other nodes [22]. Since there is only one layer of graph network, we do not need to apply residual connections but only normalize the embeddings.

It should be noted that since the tokenizer used by PLBART is sentencepiece [23], it may break down the original words and thereby damage the graph structure. Therefore, the graph reader uses a separate vocabulary instead of sharing the same vocabulary with the pre-trained PLBART. The embedding size of the graph reader is the same as that of PLBART.

B. Encoder

Our encoder has a structure similar to that of Transformer’s encoder, except that we added another cross-attention layer after the existing self-attention layer to read grammatical information from the graph reader. This makes the structure of the GraphPLBART encoder and decoder very similar, except that the first multi-head attention layer in the encoder does not use the masking mechanism. The output of the encoder is a code embedding that has been augmented with syntactic information.
C. Decoder

Our decoder is fully based on the pre-trained PLBART decoder. Moreover, because PLBART shares the same vocabulary for encoder and decoder, we do not need to use a separate copy mechanism [3] to solve the OOV (Out Of Vocabulary) problem, which simplifies the architecture and reduces computational overhead.

IV. EXPERIMENTAL SETTINGS

A. Datasets and Evaluation Metrics

In our experiments, we utilized two public datasets that are commonly used. The first dataset is the Java dataset [5], which contains 87,136 Java code snippets with accompanying comments written by developers. The second dataset is the Python dataset [24], which encompasses 87,226 Python code snippets along with comments. To ensure fairness during comparison, we employed the same pre-divided datasets as [4]. We utilize three commonly used metrics to evaluate our model: BLEU [25], METEOR [26], and ROUGE-L [27].

B. Baseline

We compared the commonly used model and the proposed GraphPLBART model in this paper using data from [28].

- **Hybrid2Seq** [24] is a deep reinforcement learning framework that utilizes an LSTM-based encoder to learn from code snippets and binary trees generated from ASTs.
- **DeepCom** [5] uses an LSTM-based architecture to enhance the quality of comments by analyzing their structure and converting ASTs into token sequences.
- **API+Code** [29] learns API knowledge from source code API sequences and applies the learned features to enhance code summarization performance.
- **Dual Model** [30] uses a dual training framework to train code generation and code summarization tasks together. It employs two sequence-to-sequence networks with attention to improve the performance of both tasks.
- **Transformer** [4] utilizes relative positional encoding to capture pairwise relationships between tokens in the source code text.
- **mAST+GCN** [31] combines sequential and structural features of code using AST, graph convolution, and Transformer layers for code summarization.
- **SiT** [8] utilizes a structure-induced Transformer that preserves the structural relationships and self-attention mechanism for encoding.
- **CodeT5** [16] is a pre-trained encoder-decoder model based on T5 [32] for programming and natural language, directly tuned with summarization datasets.
- **CodeBERT** [10] is a pre-trained encoder model for programming and natural language based on Roberta [33].
- **M2TS** [17] uses a multi-scale approach to extract features from ASTs, enabling more comprehensive extraction of structural information at local and global levels.

C. Hyper-parameter Setting

We build our model based on the Hugging Face transformers module, with the number of layers for GG-NN in the graph reader set to 4 while the other parameters of the model were the same as PLBART-base. During training, we set the batch size to 32, use the AdamW optimizer with an initial learning rate of 0.00005, and set the dropout to 0.1. While generating, we set the parameter num_beams to 6.

V. EXPERIMENTAL RESULTS

This section will focus on the following questions:

RQ1: How does GraphPLBART perform compared to other baselines?

RQ2: What impact does FA-AST have on the performance of GraphPLBART?

RQ3: How does GraphPLBART perform in specific examples?

A. RQ1: How does GraphPLBART perform compared to other baselines?

We present the results of our experiments, including BLEU, METEOR, and ROUGE-L scores, in Table I, where GraphPLBART and other baselines are evaluated on the testing set. The results indicate that GraphPLBART outperforms non-pre-trained models by a significant margin. In comparison to other pre-trained methods such as CodeBERT and CodeT5, GraphPLBART demonstrates superior performance in generating code summaries. For example, on the Java dataset, GraphPLBART achieves a top BLEU score of 47.56, while

---

3https://huggingface.co/docs/transformers/index

---
M2TS and CodeBERT achieve slightly lower scores at 46.84 and 46.64, respectively. Additionally, GraphPLBART outperforms other baseline models in the Meteor metric. However, its Rouge-L score is slightly lower than M2TS, which may be attributed to the fact that the graph network ignores the order of nodes and thus has a certain impact on the summary’s order.

It is worth noting that the performance of GraphPLBART differs significantly from other baseline models on the Python dataset. This discrepancy can be attributed to the small size of the dataset, which hinders GraphPLBART’s ability to fully converge on the pre-trained parameters. Nonetheless, these findings confirm the effectiveness of GraphPLBART in code summarization tasks and highlight the importance of pre-training for optimal performance.

B. RQ2: What impact does FA-AST have on the performance of GraphPLBART?

To investigate the role of graphs in code summarization, we conducted an experiment where we removed the graph reader from GraphPLBART as well as the multi-head self-attention layer that was added in the encoder. This effectively restored GraphPLBART to its original form, PLBART. The results of this experiment are shown in Table I. We found that GraphPLBART outperformed PLBART (GraphPLBART w/o graph) in all three metrics for Java datasets. This suggests that the fine-grained structure and semantic information provided by FA-AST is crucial during the inference process. However, for Python datasets, we observed that the non-graph network performed significantly better than GraphPLBART and other baselines. We suspect that this is because all parameters in PLBART without a graph were pre-trained, which greatly reduced the requirement for additional training data. On the other hand, the Python dataset was too small for the graph reader to fully converge. This finding confirms the importance of pre-training in code summarization. In summary, our experiment highlights the important role of graphs in code summarization, particularly for Java datasets.

C. RQ3: How does GraphPLBART perform in specific examples?

To further investigate the performance of GraphPLBART, we examine the summaries generated by it for some given codes, as shown in Fig 3. As we can see, the model without a graph is unable to pay attention to the IF control statement, resulting in the lack of premise conditions in the generated summary. However, GraphPLBART can notice it. In the Python example, isdigit is a method of the str class in Python. GraphPLBART successfully captures this syntax information and provides the string in the summary, while PLBART can only provide the variable name text while inferencing.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Java BLEU</th>
<th>Java METEOR</th>
<th>Java ROUGE-L</th>
<th>Python BLEU</th>
<th>Python METEOR</th>
<th>Python ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>API+CODE (2018) [29]</td>
<td>41.31</td>
<td>23.73</td>
<td>52.25</td>
<td>15.36</td>
<td>8.57</td>
<td>33.65</td>
</tr>
<tr>
<td>Transformer (2020) [4]</td>
<td>44.58</td>
<td>26.43</td>
<td>54.76</td>
<td>32.52</td>
<td>19.77</td>
<td>46.73</td>
</tr>
<tr>
<td>mAST+GCN (2021) [31]</td>
<td>45.49</td>
<td>27.17</td>
<td>54.82</td>
<td>32.82</td>
<td>20.12</td>
<td>46.81</td>
</tr>
<tr>
<td>SIT (2021) [8]</td>
<td>45.19</td>
<td>27.52</td>
<td>55.87</td>
<td>34.31</td>
<td>22.09</td>
<td>49.71</td>
</tr>
<tr>
<td>CodeTS (2021) [16]</td>
<td>46.01</td>
<td>28.55</td>
<td>56.49</td>
<td>34.31</td>
<td>22.09</td>
<td>49.25</td>
</tr>
<tr>
<td>CodeBERT (2020) [10]</td>
<td>46.64</td>
<td>28.84</td>
<td>56.23</td>
<td>34.34</td>
<td>21.99</td>
<td>49.73</td>
</tr>
<tr>
<td>M2TS (2022) [17]</td>
<td>46.84</td>
<td>28.93</td>
<td>57.87</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GraphPLBART w/o graph</td>
<td>47.26</td>
<td>30.71</td>
<td>56.91</td>
<td>34.94</td>
<td>23.29</td>
<td>50.55</td>
</tr>
<tr>
<td>GraphPLBART</td>
<td>47.56</td>
<td>30.95</td>
<td>57.57</td>
<td>33.81</td>
<td>21.56</td>
<td>46.26</td>
</tr>
</tbody>
</table>

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduce a novel graph augmentation approach for code summarization, which effectively leverages structural information in a pre-trained model. We conducted experiments to evaluate the effectiveness of our GraphPLBART model and found that it outperforms baseline models...
in certain metrics. Our work highlights the potential of pre-trained models for programming languages in code summarization tasks. In future work, we aim to explore more advanced pre-training tasks to capitalize on the large-scale training data, particularly for pre-training graph models.

ACKNOWLEDGMENT
This work is partially supported by the National Natural Science Foundation of China (62172202), Collaborative Innovation Center of Novel Software Technology and Industrialization, the Major Program of the Natural Science Foundation of Jiangsu Higher Education Institutions of China under Grant Nos.22KJA520008, the Priority Academic Program Development of Jiangsu Higher Education Institutions, and the Undergraduate Training Program for Innovation and Entrepreneurship, Soochow University(202210285197H).

REFERENCES
Session KE: Knowledge Engineering
An Efficient Lossless Graph Summarization Method for Large Streaming Graphs

Qia Wang, Yi Wang, Ying Wang
College of Computer and Information Science College of Software
Southwest University
ChongQing, China
1159838035@qq.com, echowang@swu.edu.cn, waying95@swu.edu.cn

Abstract—Graph summarization aims to extract critical information from large graphs by creating summaries that represent the original data. Especially, real-world daily applications generate massive dynamic streaming graphs, representing as edge or node streams. How to efficiently generate compact and lossless graph summaries for large streaming graphs is still a challenging problem. This paper proposes an efficient and scalable lossless summarization method for streaming graphs, called Partition and Similarity-based Two-Stage Summarization (PSTSS). PSTSS uses a streaming graph partitioning algorithm as the first stage to generate coarse-grained summaries by on-the-fly subgraph partitioning. The second stage generates fine-grained supernodes within these coarse-grained supernodes through efficient node similarity calculation. Our experiments on six large datasets demonstrate that our method achieves better operational efficiency, compression rate, scalability, and graph query speed compared to state-of-the-art lossless summarization algorithms.

Index Terms—graph summary; graph compression; complex network; streaming graph; graph partition

I. INTRODUCTION

Graphs are applied to represent interconnected data for the applications including web graphs, social networks, communication networks, citation networks, and even protein-protein interactions. With the emergence of big data, the sizes of real-world graphs are growing at an unprecedented rate. For instance, as of June 30th, 2022, WeChat has reached a scale of billions with 1.299 billion monthly active accounts [1]. The World Wide Web’s indexed web pages have surpassed at least 6.42 billion [2] by November 2022. Moreover, large graphs generated by the daily activities of real-world applications are normally dynamic graph streams, representing as edge streams or node streams. Therefore, efficient storage, querying and visualization of large streaming graphs is an urgent problem that needs to be solved to better support downstream applications.

Graph summarization is a crucial technique for representing large graphs in a concise manner, which is a trending topic in data mining. Most previous works have focused on static graphs [3]–[8]. These static graph summarization methods require importing the entire graph into memory for summarization which results in high memory requirements and poor scalability. Existing streaming graph summarization methods [9]–[13] generate lossy summaries and cannot achieve the accurate reconstruction of the original graph. Moreover, in incrementally summarization, they mainly consider the case of directly adjacent nodes. They lack recognition of the features of nodes in a more comprehensive local range, which affects the compression ratio and readability of the graph summary.

To address the shortcomings of the above graph summarization methods, this study proposes PSTSS (Partition and Similarity-based Two-Stage Summarization), an efficient lossless graph summarization method for large-scale streaming graphs. Our contributions are as follows:

- **Efficient Lossless Graph Summarization Algorithm** We propose PSTSS for lossless graph summarization. With linear scalability, PSTSS summarizes large-scale graph faster and achieves better compression than the state-of-the-art graph summarization method.

- **Query Evaluation Algorithm for PSTSS Summary** We propose neighborhood queries evaluation algorithm for PSTSS Summary. Neighborhood queries can be answered quickly from a summary graph and edge corrections.

- **Extensive Experiments** We confirmed that PSTSS outperforms 3 state-of-the-art graph summarization algorithms on 6 real graphs in different domains.

II. RELATED WORK

Graph summarization is a known NP-hard problem [14]. Previous research has extensively explored state-of-the-art methods in graph summarization, including a comprehensive review of existing algorithms [15], [16]. This section specifically discusses lossless and lossy summarization techniques as well as streaming graph summarization, which are related to our study closely.

A. Lossless Summarization

The goal of lossless summarization is to identify the most concise summary graph \(G^* = (S, P)\) that can accurately recreate the initial graph. Navlakha et al. [3] were the pioneers in solving the problem of lossless graph summarization using the minimum description length (MDL) principle [17]. They introduced two heuristics, Randomized and Greedy, to
minimize the summary graph and edge corrections through random selection and greedy strategies respectively. Khan et al. [4] employed unified locality sensitive hash (ULSH) to quickly select node pairs for merging, requiring a relatively high computational cost. Shin et al. [5] grouped node pairs using min-hash before merging highly similar nodes in each group. Ko et al. [9] on the other hand, maintain a streaming graph summary incrementally in response to edge additions and deletions while ensuring compression ratio and efficiency.

B. Lossy Summarization

The goal of lossy graph summarization is to create a brief summary graph that preserves the neighbors of each node in the original decompressed graph. Two algorithms, APXMDL [3] and SWEG-lossy [5], are used for this purpose. Both algorithms take edge corrections into account in their output results to minimize the most concise representation $G^* = (S, P, C^+, C^-)$ of the original graph. Other algorithms, such as K-GS [6], S2L [7], and SSUMM [8], represents the summary graph as $G^* = (S, P)$ without considering edge corrections. The K-GS algorithm selects node pairs from a pool of candidate nodes and merges them repeatedly to decrease the adjacency matrix of both the input graph and its reconstructed output representation. Meanwhile, S2L guarantees an approximate "p-reconstruction error" for the input graph by utilizing geometric clustering to summarize it. Finally, SSUMM applies the MDL principle to balance size and accuracy of summarising by sparsifying the original graph for lossy summarization purposes.

C. Streaming graph Summarization

Streaming graph summarization is more challenging than static graph summarization due to the dynamic of streaming data. Algorithms of streaming graph summarization mainly use statistical methods to process and analyze basic structures in the streaming graph, such as estimating edge frequency and node degree distribution. For example, MoSSo groups similar nodes and incrementally calculates a lossless summary, while gSketch [19] estimates edge frequency to generate a lossy summary that supports structural queries. GS4 [10] generates a lossy summary using the sliding window model and vertex properties of the graph stream. In literature [11], a compressed binary tree corresponds to the streaming graph data for lossy summarization. In literature [12], hash functions maintain a minimum neighborhood sample subgraph in real-time. GSS [13] first generates a sketch of the streaming graph using hash functions, then uses a novel data structure to store it, achieving lossy summarization supporting various queries.

Mainly of the streaming graph summarization algorithms mentioned above are lossy and focus primarily on directly adjacent nodes. However, they fail to recognize the features of nodes within a broader local range, which can impact both compression ratio and readability of the graph summary.

III. TWO-STAGE LOSSLESS SUMMARIZATION METHOD FOR STREAMING GRAPH

A. Overview of PSTSS

PSTSS adopts a divide-and-conquer algorithm to partition the streaming graph online and generate coarse-grained summary by dividing it into multiple subgraphs. Additionally, fine-grained supernodes are generated through efficient node similarity calculation. Fig. 1 illustrates the two-stage summarization process. Firstly, a greedy algorithm partitions the streaming graph [20], [21] to generate a coarse-grained summary of the graph. Secondly, each coarse-grained supernode is summarized separately using a node similarity algorithm to obtain fine-grained supernodes along with their corresponding sets of superedges and edge corrections. This lossless summarization method enables reconstruction of the original graph.

B. Streaming Graph Summary Problem Description

This study focuses on streaming (undirected) graphs represented as node streams, which consist of a sequence of nodes with their corresponding neighbor lists. Alternatively, streaming graphs can also be represented as edge streams, but for the purposes of this study, they will not be considered. Given a node streaming graph $G_t = (v_t, N(v_t))_{t=0}^{\infty}$, where $v_t$ is a node and $N(v_t)$ is the set of neighboring nodes of $v_t$. $G = (V, E)$ is a snapshot of the streaming graph at a given moment, and $V$ and $E$ are the set of nodes and the set of edges of the streaming graph, respectively. The lossless summary of $G$ is $G^* = (S, P)$ and the set of edge corrections $C$, where $S$ is the set of supernodes, $P$ is the set of supernodes, and $C$ is the set of edge corrections $C = (C^+, C^-)$.

For a given streaming graph $G$, the goal of lossless graph summarization is to efficiently generate a compact summary that can be used to reconstruct the original graph without losing any nodes or edges.

C. Coarse-grained Summary Method Based on Greedy Partitioning Algorithm

This study uses a streaming graph partitioning algorithm to partition the streaming graph and create a coarse-grained
Algorithm 1 Coarse-grained summary of streaming graph

Input: \( G_i \) and \( eC \)

1: for \( i = 1 \) to \( i = k \) do
2: \( w_i = 1 - \frac{|Pa_i|}{Ca} \)
3: \( g(v, Pa_i) = |Pa_i \cap N(v)| + w(i) \)
4: end for
5: for \( i = 1 \) to \( i = k \) do
6: \( ptno = \text{argmax}(g(v, Pa_i)) \)
7: end for
8: \( P_{ptno} \leftarrow P_{ptno} \cup v_i \)
9: \( C.add(E_i) \)
10: Return \( G_i \) and \( eC \)

### Table I

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N(v) )</td>
<td>neighborhood of node ( v )</td>
</tr>
<tr>
<td>( C = (C^+, C^-) )</td>
<td>edge corrections</td>
</tr>
<tr>
<td>( C^+ )</td>
<td>set of edges to be inserted</td>
</tr>
<tr>
<td>( C^- )</td>
<td>set of edges to be deleted</td>
</tr>
<tr>
<td>( k )</td>
<td>number of partitions, ( k \in N )</td>
</tr>
<tr>
<td>( Pa_i )</td>
<td>a set of vertices in a partition ( i ), ( i \in [1, k] )</td>
</tr>
<tr>
<td>( G_i = (Pa_i, E_i) )</td>
<td>subgraph after graph partition</td>
</tr>
<tr>
<td>( \pi = {G_1, G_2, \ldots, G_k} )</td>
<td>a set of subgraphs (Coarse-grained Summary)</td>
</tr>
<tr>
<td>( \pi(v, Pa_i) )</td>
<td>supernode in ( \pi ) that contains node ( v )</td>
</tr>
<tr>
<td>( G^* = (S, P) )</td>
<td>a summary graph with supernodes ( S ) and superedges ( P )</td>
</tr>
</tbody>
</table>

Table of Symbols

- \( \pi \): Coarse-grained summary of streaming graph (Algorithm 1)
- \( eC \): Set of edge corrections (Algorithm 2)
- \( G \): Graph
- \( V \): Set of nodes
- \( E \): Set of edges
- \( Pa \): Partition
- \( C \): Capacity
- \( \pi \): Summary
- \( \pi(v) \): Set of nodes in the summary
- \( \pi(v, Pa) \): Supernode in \( \pi \) that contains node \( v \)
- \( G^* \): Fine-grained summary
- \( S \): Set of supernodes
- \( P \): Set of superedges

Summary. The algorithm takes a sequence of nodes with their neighbor lists as input (node stream) and partitions them into subgraphs based on load balancing principles while minimizing edge cuts. Unlike traditional graph partition algorithms, this process only requires one graph traversal, making it suitable for large-scale graphs with lower time complexity. References include [22, 23]. For an undirected graph \( G = (V, E) \), the node division of the graph divides the nodes in the graph \( G \) into \( k \) partitions, each of which is denoted by \( Pa_i(i \in \{1, \ldots, k\}) \) and satisfies the following three conditions: 1) \( Pa_i \neq Pa_j(i, j \in \{1, \ldots, k\}, i \neq j) \); 2) \( \cup Pa_i = V; 3) \emptyset \neq Pa_i \subseteq V \).

**Definition 1 Coarse-grained summary of streaming graph**

Given a node streaming graph \( G_i = (v_i, N(v_i))_{i=0}^{\infty} \), the subgraph after division, and \( eC \subseteq E \) is the set of edge cuts resulting from the division of the set of nodes. \( \pi \) and \( eC \) are the coarse-grained summary of the streaming graph.

Algorithm 1 is given the number of divisions \( k \). Firstly, the original graph is read in real-time as a sequence of nodes and their neighborhood; Secondly, under the premise of satisfying load balancing, the nodes are divided according to equations (1) and (2), and the read nodes are divided into partitions that maximize the value of Equation (1); Finally, the coarse-grained summary is calculated, and the edges connecting each subgraph are saved as edge corrections.

\[
g(v, Pa_i) = |Pa_i \cap N(v)| \left(1 - \frac{|Pa_i|}{Ca}\right) \quad (1)
\]

\[
f(v) = \text{argmax}_{i \in [1,k]} \{g(v, Pa_i)\} \quad (2)
\]

In Equation (1), the partition capacity \( Ca = |V|/k \), node division needs to ensure the load balancing degree to achieve the division of the nodes of the balanced graph as much as possible. \( g(v, Pa_i) \) denotes the value of load balancing degree with \( |Pa_i \cap N(v)| \), and \( f(v) \) in Equation (2) denotes the division of node \( v \) into partitions that maximize the value of \( g(v, Pa_i) \).

Algorithm 1 computes a coarse-grained summary of the resulting streaming graph based on the greedy partitioning algorithm described above. Let the number of partitions be \( k \). Algorithm 1 traverse every node in the graph, so the space complexity is \( O(V) \) and the time complexity is \( O(k|V|+|E|) \), where \( V \) and \( E \) are the set of nodes and the set of edges of the graph at the current moment.

**Example 1** Be provided with an undirected graph \( G \) shown in Fig.1 containing 13 nodes and 11 edges, and determine the number of partitions \( k = 2 \). Under the premise of satisfying the capacity constraint \( Capacity = 11/2 = 6 \), \( g(v, Pa_i) \) is computed separately for all possible partitioning cases into two subgraphs, then the maximum value of \( g(v, Pa_i) \) is selected among all partitioning results, which is the two subgraphs \( G_1 \) and \( G_1 \) for graph partitioning. The process of graph partitioning generates edge cut sets, i.e., edge \( \{a, b\} \) and edge \( \{a, c\} \), which are saved as edge cut set \( eC \) supports the reconstruction of the graph.

**D. Fine-grained Summary Method Based on Node Similarity**

This study uses coarse-grained summarization to perform fine-grained supernode calculation. The method is based on node similarity, which groups nodes by identifying their structural similarities. This allows a supernode to replace a group of nodes in the resulting fine-grained summary.

**Definition 2 Fine-grained summary of streaming graph**

Given a node streaming graph \( G_i = (v_i, N(v_i))_{i=0}^{\infty} \), the coarse-grained summary is obtained as \( \pi \) and \( eC \). And for each subgraph, \( G_i = (Pa_i, E_i)(i \in \{1, \ldots, k\}) \) in the coarse-grained summary, the nodes are merged according to the node similarity to form the fine-grained supernodes. The superedges and edge corrections are generated to obtain the fine-grained summary of the streaming graph \( G^* = (S, P) \) and edge corrections \( C \).

Algorithm 2 describes the fine-grained summary method. Algorithm 2 is divided into three main tasks: 1) calculating node pairs to compute similarity and sorting node pairs by similarity; 2) merging highly similar nodes; 3) connecting supernodes to form super edges and edge correction.

**Selection of Fine-grained Supernode Candidates** When dealing with large graphs that have node bases of orders of magnitude, evaluating all possible pairs of directly or indirectly connected nodes to determine supernodes is inefficient. To address this issue, PSTSS introduces the locality-sensitive
hashing (LSH) method which accurately identifies similar nodes. LSH calculates similarity by comparing hash codes of each node’s neighborhood. Nodes falling into different buckets are considered dissimilar and their similarity is set to 0; these node pairs are not merged. This approach limits the computation of node similarity to only those in the same bucket, significantly reducing computational resources and running time while still identifying similar nodes.

**Algorithm 2 Fine-grained summary of streaming graph**

**Input:** $\pi, eC, h(HashFunction), \tau(\text{SimilarityThreshold})$

**Output:** $G^* = (S, P), C$

1. for $G_i \in \pi$ do
2. for $v \in G_i$ do
3. Create minhash signature column, $MSC_v$, using neighbors of $v$ by $h$ hash functions
4. end for
5. for $G_i \in \pi$ do
6. compute $Sim(v, u)$ for each bucket
7. $Sim(v, u) \leftarrow 0$ for $v$ and $u$ in different buckets
8. Rank AdaSim scores to generate $F$
9. Pop $v, u$ from $F$
10. if $Sim(v, u) > \tau$ then
11. merge $v, u$ and update $S$
12. end if
13. end for
14. end for
15. for $A \in S$ do
16. for $B \in S$ do
17. if $E_{AB} \neq 0$ and $|E_{AB}| \leq \left(\frac{|T_{AB}| + 1}{2}\right)$ then
18. $C^+ \leftarrow C^+ \cup E_{AB}$
19. else
20. $P = P \cup A, B$
21. $C^- \leftarrow C^- \cup (T_{AB} - E_{AB})$
22. end if
23. end for
24. end for

**Method of calculating node similarity** Algorithm 2 calculates the similarity of node pairs using the recursive similarity measure in AdaSim [24]. In some of the existing similarity calculation methods, only the common neighboring nodes directly connected to the selected node pair are often considered; In this method, indirectly connected nodes are also considered. Equations (3) and (4) give the formulas for calculating the iterative form of AdaSim. When $l = 0$, $Sim_0(a, b) = 1$ if $a = b$, and $Sim_0(a, b) = 0$ if $a \neq b$. $D \in (0, 1)$ is the determining damping factor, the parameter $\theta \in (0, 1]$ is an important parameter, and $m$ is the maximum value of Ada similarity. First, the weights of the common neighbor nodes directly connected to nodes $a$ and $b$ (with path length 1) are summed and calculated to obtain the similarity $Sim_1(a, b)$ in the first step. Next, the common nodes indirectly connected with nodes $a$ and $b$ at a distance of path length two are iteratively computed. To balance the computational efficiency and similarity accuracy, this study considers the node similarity within two steps, i.e., $Sim_2(a, b)$ is used as the similarity of nodes.

$$Sim_l(a, b) = D \times \left(\frac{\theta}{m} \sum_{i \in N(a) \cap N(b)} wt_i + \sum_{\tau \in N(a) \cap N(b)} wt_i \times \sum_{i \in N(a) \cap N(b)} wt_i \times wt_j \times \frac{1 - \theta}{\sum_{\tau \in N(a) \cap N(b)} wt_i \times \sum_{\tau \in N(a) \cap N(b)} wt_i} \right)$$ \hspace{1cm} (3)

$$wt_i = \frac{1}{\log(|N(i)| + e)}$$ \hspace{1cm} (4)

After calculating the similarity of node pairs, the node pairs with high similarity need to be merged. The node pair sequence $F$ is obtained by fast sorting according to the likeness of node pairs, setting the similarity threshold $\tau$, and selecting node pairs greater than $\tau$ from the node pair sequence for merging according to the similarity. After generating the supernodes, the method connects the supernodes to form superedges and edge corrections according to the best encoding.

**Optimal encoding** For each supernode pair $\{u, v\}$, let $E_{AB} = \{\{u, v\} \in E | u \in A, v \in B, u \neq v\}$ and $T_{AB} = \{\{u, v\} \subseteq V | u \in A, v \in B, u \neq v\}$ be the set of existing and potential edges between $A$ and $B$, respectively. The edges between $A$ and $B$ (i.e., $E_{AB}$) are encoded according to the following rules.

1. If $|E_{AB}| \leq \left(\frac{|T_{AB} + 1|}{2}\right)$, then add all edges in $E_{AB}$ to $C^+$;
2. If $|E_{AB}| > \left(\frac{|T_{AB} + 1|}{2}\right)$, then add the superedge $A, B$ to $P$ and $T_{AB} \setminus E_{AB}$ to $C^-$.

**Complexity Analysis** Algorithm 2 is divided into three main tasks. The time complexity of task 1 is $O(E)$, and the space complexity is $O(V)$, where $E$ is the base of the set of edges in the original graph and $V$ is the base of the set of nodes in the original graph; Task 2 first applies the LSH algorithm in the graph with time complexity of $O(E)$, and then calculates the similarity and sorts the node pairs for each bucket with space complexity of $O(V)$. The time complexity of sorting the sequence of node-pairs is $O(E + LogF)$, where $F$ is the number of node pairs; The time complexity of task 3 is $O(E)$, and the space complexity is $O(V)$. In summary, the space complexity of Algorithm 2 is $O(V)$, and the time complexity is $O(E + LogF)$.

**Example 2** The undirected graph $G$ shown in Fig. [1] is divided to generate a coarse-grained summary, and then two subgraphs, $G_1, G_2$, and edge correction $C$ are formed. Then the nodes are fine-grained summarized according to similarity. Firstly, the LSH algorithm is used to group the nodes and calculate the AdaSim similarity, merge the highly similar nodes to form the supernodes and connect the superedges, and finally obtain the summary graph $G^* = (S, P)$ and edge correction $C$. Note that in the process of grouping the nodes to form the supernodes, unlike calculating the similarity within only one step, this study considers the nodes at a
distance of two steps to calculate the parallel, which more accurately captures the local similarity features of the nodes. For example, nodes \( e, j, h, k \) are merged into one supernode based on the high similarity of AdaSim within two steps.

E. Query Evaluation Algorithm for Summary Graph

Neighborhood queries are a key building block that is reused in many graph algorithms (Dijkstra algorithm, PageRank, ShorestPaths, etc.). Algorithm 3 describes how to answer neighbor queries (given a node \( v \in V \), find the neighbors \( N(v) \) of \( v \) on a summary graph \( G^* = (S, P) \) and edge correction \( C = (C^+, C^-) \) without reconstructing the original graph. Let the neighbors of node \( v \) in edge corrections \( C^+ \) be \( N^+(v) \) and the neighbors of node \( v \) in edge corrections \( C^- \) be \( N^-(v) \).

Algorithm 3 Neighbor Query Processing on PSTSS summary

Input: summary graph \( G^* = (S, P) \), edge corrections \( C = (C^+, C^-) \), query node \( v \in V \)

Output: \( N(v) \)

1: if \( S(v) \) has a self-loop in \( G \) then
2: \( N(v) = N(v) \cup (S(v) - \{v\}) \)
3: end if
4: if for each neighbor \( A(\neq S(v)) \) of \( S(v) \) in \( G^* \) then
5: \( N(v) = N(v) \cup A \)
6: end if
7: \( N(v) = (N(v) \cup N^+(v)) - N^-(v) \)
8: Return \( N(v) \)

IV. EXPERIMENTS

A. Experimental Settings

Experimental environment: Intel Core i5 and 8 GB main memory, having 64 bit Windows 10 Professional edition.

Comparison algorithms: Experimentally, three state-of-the-art algorithms are selected from the lossless graph summarization algorithms as comparison algorithms. (1) SAGS \( (h=30, b=10, p=0.3) \) (2) SWeG \( (T=10, e = 0) \) (3) MoSSo \( (e = 0.3, c = 120) \).

B. Datasets

To verify the effectiveness of PSTSS, six large graph datasets from different domains (Table II) were selected for the experiments of running efficiency and compression rate comparison. In every graph in Table II the experiment ignores the direction of all edges and removed both self-loops and multiple edges.

C. Speed

The experiment first compared the running time of the PSTSS algorithm with the lossless graph summarization algorithms SAGS, SWeG, and MoSSo. The specific running time comparison results are shown in Table III. With the same datasets selected for graph summarization, the PSTSS algorithm proposed in this paper has higher summarization efficiency than the other three lossless graph summarization algorithms on all six datasets. Among them, the PSTSS algorithm significantly outperforms the other lossless summarization methods on the dataset Ego-Facebook.

D. Compression Ratio

To verify the effectiveness of the PSTSS algorithm, the experimental uses the compression ratio defined in Equation (5) to measure the relative size of the summary graph to the original graph. With the same original graph, the smaller the compression ratio value, the more compact the output graph is proved to be, and the more influential the algorithm is in compressing the original graph. The experiments were performed to summary the datasets in Table III by three comparison algorithms and PSTSS, and the specific experimental results are shown in Fig. 2.

\[ RN = |S|/|V| \]  \hspace{1cm} (5)

As shown in Fig. 2, the compression ratio of the proposed summarization method PSTSS algorithm in this study outperforms all three comparison algorithms, SAGS, SWeG, and MoSSo.

E. Scalability

To measure the scalability of the PSTSS algorithm, experiments were designed to test the variation of the running time of the PSTSS algorithm on datasets of different sizes. Experiment sampled different numbers of nodes from the LiveJournal dataset and generated multiple graph datasets and used them as input graphs. As shown in Fig. 3, the running time of the PSTSS algorithm increases linearly with the number of nodes in the input graph with a slope close to 1, which indicates that the PSTSS algorithm has great scalability.

F. Neighbor queries on query PSTSS summary

The summary results generated by the PSTSS algorithm can significantly reduce the query time to the original graph. To verify this property of the PSTSS algorithm, experiments on neighbor node queries are designed in this study. The experiments were performed using the results of the PSTSS...
algorithm in the dataset of Table [II] for neighbor queries, running Algorithm 3 and comparing the queries times with the SWeG algorithm. Fig. 3 shows the reduction in running time.

V. CONCLUSION

This study proposes a two-level lossless summarization method called PSTSS to address the challenge of large streaming graph summarization. The method is based on streaming graph partitioning and node similarity. Results from experiments conducted on six large graph datasets demonstrate that compared to three typical lossless graph summarization algorithms, PSTSS has lower time complexity, better compression rate, higher operational efficiency, good scalability, and can improve the query efficiency of graphs. It can effectively implement large graph summarization.

Future research directions include exploring parallel streaming graph summarization and combining stream graph summarization methods with techniques such as graph visualization and mining.

REFERENCES

Combining Structure Embedding and Text Semantics for Efficient Knowledge Graph Completion

Wen Sun
School of Informatics
Xiamen University
Xiamen, China
wensun.cs@gmail.com

Yifan Li
School of Film
Xiamen University
Xiamen, China
liyifan422@stu.xmu.edu.cn

Junfeng Yao
School of Film
Xiamen University
Xiamen, China
yao0010@xmu.edu.cn

Qingqiang Wu*
School of Informatics
Xiamen University
Xiamen, China
wuqq@xmu.edu.cn

Kunhong Liu*
School of Film
Xiamen University
Xiamen, China
lkhqz@xmu.edu.cn

Abstract—Knowledge graph completion plays a crucial role in downstream applications. However, existing methods tend to only rely on the structure or textual information, resulting in suboptimal model performance. Moreover, recent attempts to leverage pre-trained language models to complete knowledge graphs have proved unsatisfactory. To overcome these limitations, we propose a novel model that combines structural embedding and semantic information of the knowledge graph. Compared with previous works based on pre-trained language models, our model can better use the implicit knowledge of pre-trained language models by using relation templates, entity definitions, and learnable tokens. Furthermore, our model employs a multi-head attention mechanism to transform the embedding semantic space of entities and relations obtained from the knowledge graph embedding model, thereby enhancing their expressiveness and unifying the semantic space of both types of information. Finally, we utilize convolutional neural networks to extract features from the matrices created by combining these two types of information for link prediction and triplet classification tasks. Empirical evaluations on two knowledge graph completion datasets demonstrate that our model is effective for both tasks.

Index Terms—Knowledge graph completion, Knowledge graph, Link prediction

I. INTRODUCTION

A knowledge graph (KG) is a structured representation of the objective world that captures information about objects and their relations, typically composed of fact triples that describe the relationships between head entities and tail entities [1]. KGs have a significant impact on various natural language processing tasks [2], such as question answering, information retrieval, and recommendation systems. However, the challenge of the incompleteness problem in KGs has long impeded their effectiveness in various downstream applications.

To address this problem, researchers have turned to knowledge graph completion (KGC) methods, which aim to predict missing entities or relations in factual triples. Building on the success of word embeddings in capturing semantic information, knowledge graph embedding (KGE) models have been developed to address the link prediction problem [3], [4]. These methods treat entities and relations as continuous low-dimensional embeddings that can effectively preserve the semantics and intrinsic structure of entities and relations, allowing for computable representations. By fully comprehending the existing structures in the knowledge graph, the KGE models achieve missing link prediction by designing corresponding scoring functions and learning low-dimensional continuous vector representations of entities and relations. This methodology enables effective knowledge graph completion by accurately predicting missing entities or relations.

KGE has become the most popular method for KGC due to its simplicity and efficiency. However, this approach is limited to using structural information from existing KGs and is not effective for predicting entities and relations that are not present in the training set, thus making it less suitable for completing sparse knowledge graphs. Therefore, it is crucial to incorporate relevant textual information, such as entity and relation definitions and descriptions, to enrich the representation vector [5], [6].

Recent advancements in deep learning have led to the development of pre-trained language models (PLMs) such as BERT [7], which have shown outstanding performance in natural language processing tasks by learning word embeddings containing rich contextual semantic information from large-scale natural language text data. As a result, using PLMs to encode text data in knowledge graphs has attracted considerable attention. However, most PLM-based KGC models [5], [8] simply concatenate entity and relation labels as model inputs, failing to take full advantage of the implicit knowledge contained within PLMs and resulting in ineffective models.

Based on the aforementioned issues, we propose SS-KGC, a novel model for KGC that integrates both structural embedding and textual semantic information. Our model utilizes PLMs to encode textual data in triples, which are then transformed into coherent sentences using relation templates. Additionally, entity definitions are incorporated to better express the semantics of the triples, and learnable tokens [11] are added to improve
the model’s effectiveness.

Furthermore, the model employs a multi-headed attention mechanism [9] to transform the semantic space of the embedding vectors of entities and relations obtained through a basic KGE model. This enhances the model’s expressive ability and unifies the space of semantic and structural information, making their integration more reasonable. Finally, the model utilizes convolutional neural networks (CNN) for feature extraction and is evaluated on link prediction and triad classification tasks [10].

Experimental results demonstrate that our model outperforms the baseline models on two benchmark test sets for KGC. In summary, our contributions are as follows:

- We propose a novel KGC model based on PLMs, which combines the structural and semantic information for KGC and has achieved excellent results in experiments.
- We experimentally demonstrate the effectiveness of our model to combine semantic information and structural embedding of knowledge graphs for KGC.

II. Method

We propose SS-KGC to combine structural embedding and text semantic information for KGC. As shown in Figure 1, SS-KGC effectively utilizes both textual and structural information from the knowledge graph for KGC. Text semantic information comes from the labels and definitions in knowledge graphs. The structural embedding is obtained by pre-training the KGE model.

A. Data Processing

As our model utilizes BERT to encode text information, it is necessary to convert the triple data format of the knowledge graph into the input format of the BERT. For the triple data, it includes meaningful entity and relation labels and entity definitions, so it needs to deal with combining entity and relation labels into sentences and adding entity definitions into them. When converting entity and relation labels into sentences, we adopt the method of designing different conversion templates for different relations to convert triples into coherent sentences [6], avoiding the problem of incoherent sentences caused by direct splicing, so that the model can better encode contextual semantic information. Table I shows the transformation templates of some relations, where [X] represents the head entity and [Y] represents the tail entity.

To fully express the semantic information of triples, we consider incorporating the definitions of entities into the model and introducing special tokens in the BERT vocabulary as learnable tokens [11]. These tokens are learnable during the training process, allowing the model to adapt to specific tasks and data by learning additional information that is not fixed. By using learnable tokens, the model can learn new representations suitable for the task when training data is limited or tasks are complex, which improves the model’s ability to generalize to new data and make accurate predictions.

B. Text Semantic Encoding

The text data of the KG is integrated into coherent sentences after passing through the data processing layer as the input of the PLM. We use the BERT [7] as the encoding model. After the tokenizer of the BERT and the data processing operations, the input of the model is as follows:

\[
S = [CLS], [SPECIAL], E_{definition}, T_{sentence}, [SPECIAL], [SEP]
\]  

Among them, the input sentence begins with the [CLS] special token and ends with the [SEP]. \(E_{definition}\) are the entity definitions, which are used to supplement the information of the head and tail entities. \(T_{sentence}\) is a special token added to the BERT vocabulary as a learnable token to make the model more effective. \(T_{sentence}\) is the sentence transformed by the designed relational template. As shown in Figure 1, taking the triple (Happy Feet, /film/film/genre, comedy) as an example, the transformed sentence is "The genre of Happy Feet is comedy."

Then, we use BERT to dynamically represent the word vector. Through its bidirectional encoding capability, the model can effectively obtain word embedding containing rich contextual semantic information. After multi-layer encoding, the sentence embedding matrix containing rich semantic information is finally obtained. Its dimension is \(d_b \times d_l \times d_k\), which \(d_b\) is the size of the data batch, and \(d_l\) is the sentence length in BERT, \(d_k\) is the word embedding dimension of the BERT.

C. Knowledge Graph Embedding

The knowledge graph embedding layer mainly designs a simple KGE model to obtain the structure embedding of entities and relations, which is used as the structural information
of the model for KGC. Based on the improvement of the TransE [3] model, we use \( f = h \times r - t \) as the KGE score function.

To enrich the expressive ability of the embedding of entities and relations, after obtaining the embeddings of entities and relations, the semantic space is transformed through the multi-head attention mechanism [9], and the semantic space of the two types of information is unified by the linear layer of the last layer. The number of heads in our experiment is 3. The self-attention is calculated as follows:

\[
Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V
\]  

(2)

Where \( Q \) is the query vector, \( K \) is the key-value vector, \( V \) is the value vector, and \( d \) is the embedding dimension. When using the multi-head attention mechanism, multiple self-attention operations are performed on the same vector sequence. Each group uses a different parameter matrix to calculate attention separately and obtain multiple outputs. The calculation formula is as follows:

\[
\text{head}_i = Attention(QW_{Qi}, KW_{ki}, VW_{Vi})
\]

(3)

Among them, \( W_{Qi}, W_{ki}, W_{Vi} \) are the \( i \)-th mappings. These outputs are finally concatenated together.

\[
\text{Multihead} = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W_o
\]

(4)

Among them, the parameter matrix \( W_o \) is used to unify the semantic space output by the independent attention mechanism.

D. Convolutional Neural Network Layers

After obtaining both types of information, we utilize a convolutional neural network layer for further feature extraction in downstream tasks such as link prediction and triple classification [10]. The key advantage of CNNs is their ability to capture local features, which are automatically combined and filtered to obtain semantic information at different levels, as reflected in the text by N-gram features. Additionally, CNNs achieve good results with faster training speeds due to their parameter-sharing property.

The CNN layer takes as input the sentence embedding matrix obtained from combining the semantic and structural information of the knowledge graph. The input matrix is convolved using 256 convolution kernels of sizes 2, 3, and 4, and then pooled using global max pooling in the pooling layer. Finally, the global feature vector is mapped to the probability distribution of the output category using a fully connected layer for classification.

E. Training

The model performs negative sampling through a combination of two negative sampling strategies [6]: (1) Randomly replace the head or tail entity of the triple with other entities to generate a triple that does not exist in the knowledge graph. (2) Using the KGE model to replace the head or tail entity with other entities with high confidence produces triples that do not exist in the knowledge graph, thereby improving the quality of negative samples. Because the latter method is relatively complex, each of these two methods provides 50% negative samples for the model. The model is trained with the cross-entropy loss function.

\[
L = - \sum_{G \in \mathcal{G} \cup \mathcal{G}'} (y_G \log(\sigma(c)) + (1 - y_G) \frac{\log((1 - \sigma(c)))}{k})
\]

(5)

Among them, \( G \) represents a triplet containing positive and negative samples, and \( y_G \) values are 0 and 1, which represent the label of the triple. \( \sigma \) is the softmax function, \( c \) represents the fully connected layer output of the final part of the model, so \( \sigma(c) \) represents the classification score of the triplet. \( k \) is a positive negative sample proportion and its value is 3 in our experiment.

III. Experiments

A. Datasets and Evaluation Metrics

We employ two sub-datasets from Wikidata and Freebase. These datasets constitute the benchmark dataset of the PKGC model [6], which is built on the basis of Wikidata and manually annotated with real negative triples to form a new dataset named Wiki27K.

The FB15k-237 dataset contains many mediator (CVT) nodes, which result in Cartesian product relations that do not make sense for the corresponding prediction tasks and inappropriately increase the accuracy of the model [12]. To address this issue, [6] created a new dataset called FB15K-237-N by removing the relations containing mediator nodes in FB15K-237. Table II shows details of the datasets. The numbers in the right two columns represent the number of triples used for evaluation under the closed world assumption (CWA) and open world assumption (OWA), respectively.

We evaluate the performance of KGC models on two tasks: link prediction and triple classification. The most commonly used indicators for link prediction are MRR and Hits@n (n is 1, 3, 10), and the final results are the average values obtained by replacing the head entity and tail entity, respectively.

The triple classification task aims to determine whether a given triple is correct, making it essentially a binary classification task. Therefore, accuracy and F1 score are used to evaluate the performance of triple classification. Our model is implemented using PyTorch based on the PKGC, with bert-base-cased used as the PLM.

<table>
<thead>
<tr>
<th>Dataset</th>
<th></th>
<th></th>
<th>Train</th>
<th>Valid</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki27K</td>
<td>27122</td>
<td>625</td>
<td>74,793</td>
<td>20,242</td>
<td>20,244</td>
</tr>
<tr>
<td>FB15K-237-N</td>
<td>11,104</td>
<td>3,106</td>
<td>87,282</td>
<td>14,082</td>
<td>16,452</td>
</tr>
</tbody>
</table>

Table II: The Statistics of Datasets.
B. Baselines

To establish a baseline for our study, we select some representative models, including TransE [3], TransC [13], ConvE [14], RotatE [4], KG-BERT [5], MTL-KGC [8], StAR [14], and PKGC [6]. When compared to PLM-based models, the original StAR and PKGC models, which use the RoBERTa-Large [16], have a large number of parameters and a high negative sampling ratio, leading to long training times. To ensure a more fair comparison of model performance, we modified the StAR and PKGC models to use bert-base-cased as PLM for the experiment, labeled as StAR (BERT-base) and PKGC (BERT-base), respectively.

C. Main Results and Analysis

Table III presents the link prediction results of our model and the baseline models on the datasets of Wiki27K and FB15K-237-K-N. Table IV lists the comparative results of these models on the triple classification task under the two datasets. All metrics are multiplied by 100. Based on the experimental results, the following conclusions can be drawn:

1. Comparing the results of the link prediction and triple classification experiments, we observe that, in the PLM-based models, the direct concatenation of entities and relations is less effective than the method of transforming triples into coherent sentences through templates. This suggests that the input format of the data has a significant impact on the model's performance. As PLMs are trained on natural language text, input data that is coherent sentences are consistent with the training data to some extent. This enables better utilization of the implicit knowledge of PLMs, resulting in sentence embedding representations that contain rich semantic information.

2. In the comparative experiments against these baseline models, our model achieved the best results. This indicates that both textual semantic information and structural information are crucial for KGC tasks. In comparison to the StAR, our model's use of pre-trained structural embeddings can effectively solve the problem of mismatch between the fine-tuning speed of the BERT and the training speed of the KGE model, leading to desirable results.

3. Compared to the PLM-based models, our model achieves desirable results by using BERT-base as PLM. This suggests that the method of transforming triples into coherent sentences can better utilize the implicit knowledge in PLMs, and performance improvement can be achieved by incorporating entity definitions and learnable tokens. Other models may require larger PLMs to obtain textual semantic information. Additionally, compared to the method of using the [CLS] vector directly to represent the semantic information of triples for classification, the feature extraction method by extracting features from the entire sentence embedding matrix using a feature extractor can achieve more effective results. This may be because the [CLS] vector does not pay enough attention to the semantics of entities and relations, while the local features obtained through CNNs focus more on the semantics of entities and relations, thereby better completing the task of KGC.

Fig. 2. The comparison of the prediction accuracy rate of the top three relations corresponding to the number of samples.

4. Compared to KGE models, PLM-based models have more advantages in triple classification tasks. This suggests that incorporating PLMs into KGC models can better determine the correctness of triples, possibly because such models use cross-entropy loss functions during training, making them more proficient in classification tasks. In addition, the results under the OWA and the CWA in triple classification tasks do not differ significantly, which may be due to the smaller ratio of negative triples containing errors obtained through negative sampling compared to link prediction tasks.

D. Case Study

Table V shows the results of some triple classification prediction cases in the FB15K-237K-N dataset. We selected cases where our model predicted correctly while PKGC (BERT-base) predicted incorrectly and analyzed them. The label represents the correctness of the triple, where 1 indicates a positive triple that exists in the knowledge graph, and 0 indicates a negative triple obtained through negative sampling. It is apparent that in the same prediction task, our model outperforms PKGC (BERT-base) in relations that require comprehensive structural information, such as the "/film/film/country" relation.

To further explore the performance of our model on various types of relations, we conducted a detailed comparative analysis experiment on the triple classification task in the FB15K-237K-N dataset. Figure 2 presents the prediction results of the model on the top three relations with the corresponding sample numbers of 2622, 1444, and 1328, respectively. The values in the figure represent the accuracy of the predictions. It can be observed that our model has more advantages in resolving complex relation patterns (1-N, N-1, and N-N), such as "/people/person/profession" and "/film/film/genre".

E. Ablation experiment

Our model incorporates learnable tokens in the data processing part. To explore the impact of this component on the model, we conducted a corresponding ablation experiment. The input data of the comparison model had the [SPECIAL] tokens removed. The comparison results are shown in Figure 3.
It shows that after removing the learnable tokens, the model’s performance drops. However, compared to other models (such as KG-BERT) that use the direct concatenation of entities and relations, our model still demonstrates very competitive results. This indicates that the form of the input data is crucial in exploiting the implicit knowledge of the PLM. Additionally, experiments demonstrate that the learnable tokens can effectively enhance the model’s performance, enabling the full mining of semantics from the triple text and better utilization of the implicit knowledge of the PLM.

To explore the impact of structural embeddings on model performance, we conducted ablation experiments on the triple classification task and analyzed the results by comparing them. Figure 4 shows that the model’s performance declines on the triple classification task without structural embeddings, whether based on the OWA or the CWA, demonstrating the significant impact of structural information on KGC tasks. It’s worth noting that our model without KGE still demonstrates very competitive results compared to other baseline models.

We also found that using CNN in our model yields better results across all metrics compared to using MLP.

### IV. Related work

#### A. Pre-trained Language Models

Pre-trained language models (PLMs) are a type of natural language processing technique based on deep learning, typically trained on large amounts of unlabeled text data [7], [16]. These models can learn the grammar, semantics, and contextual information in natural language, generating meaningful language representations. PLMs can be applied to various
natural language processing (NLP) tasks, such as text classification, named entity recognition, and machine translation, through fine-tuning or transfer learning. In recent years, PLMs have become one of the research hotspots in the field of NLP, achieving many significant experimental results.

B. Knowledge graph completion Models

Knowledge graph completion is one of the important research in the field of knowledge graph [1]. A series of methods have been proposed for KGC. Among them, methods based on representation learning have received widespread attention. It includes KGE models and PLM-based models [15]. These models aim to learn low-dimensional representations for entities and relations in the knowledge graph, enabling efficient reasoning and prediction of missing links. The most classic KGE models are the translation-based KGE models [3]. They represent each entity and relation as a low-dimensional vector and then regard the relation as the distance between the head entity and the tail entity [3], [4], [17], [18]. These models are popular because of their simplicity and efficiency.

Compared to translation-based models, semantic matching models reflect the confidence of the semantic information of the triple. It includes bilinear models and neural network models. Bilinear models use a bilinear matrix to measure the similarity between entities and relations [19], [20], while neural network models use various neural network architectures to capture the complex semantics of entities and relations [14], [21].

PLM-based KGC models are a new emerging method developed in recent years. The main idea of this model is to use the PLM to convert the textual descriptions of entities and relations into vector representations, and then use these vectors to complete missing entities or relations, such as KG-BERT [5], MTL-KGC [8], STAR [15], and PKGC [6].

V. CONCLUSION

In this paper, we propose a KGC model that combines text semantic and structural embedding to improve the performance of KGC. The model uses BERT to encode text and incorporates learnable tokens and relations templates to construct coherent sentences from triple labels and entity definitions. Our model also uses multi-head attention to transform the semantic space of the embeddings of entities and relations obtained by the KGE model. Finally, the model uses CNN to extract features from the matrices spliced by two types of information. Experimental results show that our model outperforms other baseline models on link prediction and triplet classification tasks, demonstrating its effectiveness in capturing both structural and semantic information in KGs.

ACKNOWLEDGEMENTS

This research was supported by the School of Film Xiamen University - Xiaozhi Deep Art (Xiamen) Artificial Intelligence Research Institute Co., Ltd. School of Film Xiamen University - Xiaozhi Deep Art (Xiamen) Artificial Intelligence Research Institute Co., Ltd.

REFERENCES

CRCC: Collaborative Relation Context Consistency on the Knowledge Graph for Recommender Systems

Li-e Wang*,†‡, Huachang Zeng*, Shenghan Li*, Xianxian Li*†‡, Shengda Zhuo†, Jiahua Xie‡, Bin Qu*, Zhigang Sun*†, and Tianran Liu*

* School of Computer Science and Engineering, Guangxi Normal University, Guilin, China
† Guangxi Key Lab of Multi-source Information Mining Security, Guangxi Normal University, Guilin, China
‡ Key Lab of Education Blockchain and Intelligent Technology, Ministry of Education, Guangxi Normal University, Guilin, China
§ College of Cyber Security of Jinan University, Jinan University, Guangzhou, China

{wanglie, lixx}@gxnu.edu.cn, {zhc28015, zhuosd96}@gmail.com, {lshcc9, xjhEde, qubin525, szung, ltinran}@163.com

Abstract—Knowledge graph (KG) as auxiliary information can solve the cold-start and data sparsity problems of recommender systems. However, most existing KG-based recommendation methods focus on how to effectively encode items with that users have interacted into entities and propagate them explicitly, but neglect the relation-level and context-level modeling of collaborative signals. Therefore, it is inevitable to incorporate some unrelated entities while utilizing a propagation strategy, which may weaken part of the recommendation performance.

To address this problem, we propose a novel method named Collaborative Relation Context Consistency (CRCC). Compared with other KG-based methods, we model the relation-level and context-level of collaborative signals in a fine-grained manner. Specifically, we segment the user’s collaborative knowledge graph to learn related entity information separately to enrich the embedding of users. Moreover, CRCC links the consistency score between the items that users and neighbors have interacted with as the fusion basis, and then we consider the inherent popularity of items while incorporating consistent entities to enhance the embedding representation of items. Extensive experiments on three real-world datasets show that CRCC outperforms several compelling baselines in both CTR prediction and top-K recommendation.

Index Terms—recommender systems, knowledge graph, collaborative relation, contextual knowledge

I. INTRODUCTION

Recommender systems can help users find items of interest among massive amounts of information when their demands are uncertain. It builds a user preference model using machine learning technology and makes tailored recommendations to users [1].

Existing recommendation methods can be roughly categorized into three types: collaborative filtering (CF) [2], content-based [3], and hybrid [4]. Collaborative filtering has the problems of data sparsity and cold-start, which are typically resolved by introducing some auxiliary information, such as social relations [5] or KG [6]. KG is chosen as auxiliary information since it can improve recommendation interpretability.

KG is a directed heterogeneous graph in which nodes represent entities and edges represent relationships between entities. Extensive research shows that incorporating KGs into recommendation tasks improves performance [7]–[9]. Moreover, RippleNet [10] explored the potential interests of users along the entity-relation-entity link. CKAN [11] utilized the user’s interactive items as propagation sources, obtaining the user’s potential preference and the item’s latent embedding by propagating multi-hop entities. Even though these models can iteratively propagate entities to facilitate recommendation, existing KG-based recommendation neglects the problem of collaborative relation context inconsistency due to the lack of comprehensive analysis of item type and fondness. Specifically, it can be divided into the two levels listed below:

- The first is the context level, which indicates that items’ context semantics among users in the user-item interaction graph may be different. An example of the inconsistency at the context level of items is shown in Fig. 1. Different colors represent different types of items, we can observe that \( u_3 \) will be an inconsistent neighbor of \( u_1 \). Because \( u_3 \) has interacted with item types other than clothes, such as sports and books (which may reflect greater interests), whereas \( u_1 \) interests appear to be only limited to clothes. As a result, they have quite a part of different item contexts.
- The second is the relation level. Previous work has usually focused on interactive and non-interactive relationships in user-item interaction scenarios. We make a more fine-grained distinction between interaction intentions, which are strong interactions with high rating values and...
weak interactions with low rating values, to reveal how much the user likes the item. As shown in Fig. 1, we use the solid and dashed lines to indicate the strong and weak interaction relationships, respectively. We can see that users \( u_1 \) and \( u_2 \) are neighbors and that they are both connected to the camera. However, \( u_2 \) prefers the camera (reflecting a strong interaction relationship), while \( u_1 \) is not as interested in it (a weak interaction relationship). This leads to inconsistency at the relation level because although they are neighbors and are connected to the same item, their preferences for it are not consistent.

To address these above limitations, we propose an end-to-end model named CRCC, short for Collaborative Relation Context Consistency. To enrich the user’s embedding, we segment the user’s collaborative knowledge graph into a series of sub-views and learn the related entity information separately. Moreover, CRCC considers the relation-level and context-level of user-item interaction by quantifying the consistency between the items that users and neighbors have interacted with as the fusion basis. Next, we enhance the item embedding representation by considering the item’s inherent popularity while diffusing entities with which the consistent users interacted.

Our contributions in this paper are summarized as follows:

- We propose knowledge feature learning to explore how users’ interests change with the attractiveness of inter-entity relationships.
- To the best of our knowledge, we are the first work to address the inconsistency of collaborative relation context on KG-based recommendation.
- We consider the item’s inherent popularity while diffusing entities with which the consistent users interacted.
- Extensive experiments on three public datasets show that CRCC over several convincing baselines.

II. RELATED WORK

Existing KG-based recommendation methods can be classified into three categories:

- **Embedding-based** methods used knowledge graph embedding (KGE) [12] to map entities and relations to a low-dimensional vector space. CKE [13] adopted TransR to consider the heterogeneity of nodes and relationships to extract the structural representation of items. KTUP [8] proposed a multi-task learning model that adopted TransH for recommendation tasks and knowledge graph completion. However, these models learn entity embeddings are insufficient, making them more suitable for intra-graph tasks such as link prediction.
- **Path-based** methods enriched user-item interactions by designing connection paths between entities. RuleRec [14] utilized associations between items in KG for delivering an explainable recommendation. KPRN [7] generated path representations by integrating entity and relation semantics. However, designing a meta path manually is time-consuming and laborious, especially in extremely complex knowledge graphs.
- **Unified** methods combined the embedding-based and path-based methods to propagate embedding. KGCN [15] and KGNN-LS [16] demonstrated that aggregating entity neighbor graph can improve recommendation performance. KGAT [17] proposed a collaborative knowledge graph that refines nodes’ embeddings by propagating neighbor embeddings. CKAN [11] used a heterogeneous propagation strategy to encode diverse information for a better recommendation. However, existing unified methods focus on propagating the embedding of entities, ignoring the fine-grained analysis of the relational and contextual semantics from collaborative signals.

III. PROBLEM FORMULATION

In this section, we formulate the KG-based recommendation problem as follows. In a typical recommendation scenario, we denote the sets of \( M \) users and \( N \) items by \( U = \{u_1, u_2, \ldots, u_M\} \) and \( V = \{v_1, v_2, \ldots, v_N\} \), respectively. The user-item interaction matrix \( Y \in \mathbb{R}^{M \times N} \) is determined based on the user’s implicit feedback. \( y_{uv} = 1 \) indicates that user \( u \) has interacted with item \( v \), otherwise \( y_{uv} = 0 \). In addition, \( \mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\} \) denotes the knowledge graph, where \( h, t \) represent the head entity and the tail entity, respectively. \( r \) represents the relationship between the head entity \( h \) and the tail entity \( t \); \( \mathcal{E} \) and \( \mathcal{R} \) represent the set of entities and relations, respectively. \( \mathcal{A} = \{(v, e) \mid v \in \mathcal{V}, e \in \mathcal{E}\} \), where \( (v, e) \) indicates that item \( v \) can be aligned with entity \( e \).

Given the user-item interaction matrix \( Y \) and the knowledge graph \( \mathcal{G} \), our goal is to learn a prediction function \( \hat{y}_{uv} = \mathcal{F}(u, v \mid \Theta, \mathcal{G}) \), where \( \hat{y}_{uv} \) denotes the probability that the user \( u \) will interact with items \( v \) he has not engaged with before, and \( \Theta \) denotes the model parameters.

IV. METHODOLOGY

The framework of CRCC is shown in Fig. 2, and then we elaborate on each module individually.

1. **Knowledge Feature Learning**

   We treat the user’s collaborative knowledge graph as a global view, and knowledge feature learning enriches the user preference representation by aggregating extended information containing sub-views of user interaction, which consists of knowledge feature segment and knowledge feature attentive.

   1) **Knowledge feature segment**: We segment the global view formed by user \( u \) and his interactive entity set \( I \) into a series of sub-views based on the user preference profiles. The user’s initial entity set is derived from item-entity alignment:

   \[ \mathcal{E}_u = \{v \mid (v, e) \in \mathcal{A} \text{ and } v \in \{v \mid y_{uv} = 1\}\} \quad (1) \]

   When combined with the segmented view, Eq.(1) can also be defined as follows:

   \[ \mathcal{E}_u^k = \{t \mid (h, r, t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_{u}^{k-1}\}, \quad k = 1, 2, \ldots, K \quad (2) \]

   where \( k \) indicates the \( k \)-th sub-view being segment, we then define the \( k \)-th sub-view triple set for user \( u \) as follows:

   \[ S_u^k = \{(h, r, t) \mid (h, r, t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_u^{k-1}\}, \quad k = 1, 2, \ldots, K \]

   \[ \text{It consists of entities that the user interacted with and neighboring entities.} \quad (3) \]

2. **Knowledge feature attentive**

   The framework of CRCC is shown in Fig. 2, and then we elaborate on each module individually.
2) Knowledge feature attentive: We analyze the user’s relation type preference for entities in the interaction set more thoroughly, and then try to incorporate more related entities for the relation type the user prefers:

\[ p_{ur} = \text{dynamic-fun} \left( e_{u}, e_{r} \right) \]  
\[ z_0 = \text{LeakyReLU} \left( W_0 \left( p_{ur} + e_h^i \right) + b_0 \right) \]  
\[ \pi \left( p_{ur}, e_h^i \right) = \sigma \left( W_2 \text{LeakyReLU} \left( W_1 z_0 + b_1 \right) + b_2 \right) \]

where \( e_u \) is the embedding of user \( u \), \( e_r \) and \( e_h^i \) are the embedding of relation \( r \) and head entity \( h \) for the \( i \)-th triple, respectively. \( W_r \in \mathbb{R}^{2d \times d} \) is the trainable weight matrices, \( b_r \) is the bias terms of the neural network. \( \text{dynamic-fun} \left( \cdot \right) \) is an inner product function. \( p_{ur} \) characterizes the influence degree of the relation \( r \) between entities on the user \( u \). Hereafter, we use the softmax function to normalize the coefficients across the whole triples in the triple set:

\[ \pi \left( p_{ur}, e_h^i \right) = \frac{\exp \left( \pi \left( p_{ur}, e_h^i \right) \right)}{\sum_{(r', h', t') \in S_k} \exp \left( \pi \left( p_{ur}, e_h^i \right) \right)} \]

where \( \pi \left( p_{ur}, e_h^i \right) \) controls the attentive weight generated from the user’s relation preferences and the head entity. On the user side, we build the tail entity attentive embedding \( a_{(u)}^i \) as follows:

\[ a_{(u)}^i = \pi \left( p_{ur}, e_h^i \right) e_t^i \]

where \( e_t^i \) is the embedding of tail entity \( t \) for the \( i \)-th triple.

Finally, we obtain a representation of the \( k \)-th sub-view triple set for the user:

\[ e_h^k = \sum_{i=1}^{\left| S_k^u \right|} a_{(u)}^i, \quad k = 1, 2, \ldots, K \]

where \( \left| S_k^u \right| \) is the number of triples in set \( S_k^u \).

B. Collaborative Relation Context Consistency

Distinct from previous methods that propagated users’ interactive data layer by layer to obtain latent preferences. We consider both the relation of items that users have interacted with and the consistency of their contextual semantics. Taking user \( u \) as an example, neighbors with whom \( u \) has interacted with item \( v \) are denoted as \( S(u) \), all items that user \( u \) has interacted with are defined as \( R(u) \). For user \( u \) and his interactive item \( v \), it generates a semantic query embedding \( q_{uv} \) by mapping the concatenation of user \( u \) and item \( v \) embeddings:

\[ q_{uv} = \sigma \left( W_u \left( e_u \| e_v \right) \right) \]

where \( e_u \) and \( e_v \) are the embedding of user \( u \) and item \( v \), respectively. \( \| \) denotes concatenation. Furthermore, we use the self-attention mechanism to calculate the user \( u \)'s degree of preference \( \alpha_{uv} \) for the interactive item as follows:

\[ \alpha_{uv} = \frac{\exp \left( W_k q_{uv} \| e_{uv} \right)}{\sum_{v \in R(u)} \exp \left( W_k \left( q_{uv} \| e_{uv} \right) \right)} \]

where \( e_{uv} \) is the relation embedding for user \( u \) interactive item \( v \). Analogously, for neighbor user \( p \) and his interactive items, we obtain a degree of preference \( \beta_{pv} \) for user \( p \) in the same way. Next, we consider the consistency between the user and his neighbors and the items with that they have interacted. The final consistency score \( \gamma_{up} \) is defined as follows:

\[ \gamma_{up} = W_d \cdot \sigma \left( W_e \cdot \text{fusion} \left( \alpha_{uv} e_{uv}, \beta_{pv} e_{pv} \right) + b_e \right) + b_d \]

where \( \text{fusion}( \cdot, \cdot) \) is a inner product function. Normalizing Eq.(12) by the softmax function yields the consistency score between the two users. We carefully tune this consistency score \( \gamma_{up} \) to a reasonable threshold and find that a threshold higher than about 0.6 is better for filtering out users with high consistency scores, which also means that some of the noise caused by aggregating entities to multiple orders is diminished. \( \mathcal{U}_{\text{consis}} \) will be denoted as the set of consistent users.

C. Knowledge Structure-aware

Different users express various degrees of fondness for an item based on its inherent popularity. Knowledge structure-
aware enhances item embedding by accounting for the item’s inherent popularity while also improving the representation of consistent users’ interactive entities, which consists of knowledge structure diffusion and knowledge-aware attentive.

1) Knowledge structure diffusion: It considers the inherent popularity of an item while iteratively diffusing user interactive entities. On the item side, the mapping of items interacted by $U_{consis}$ with a high consistency score into entities is defined as follows:

$$V_u = \{v_u | u \in \{u | y_{uv} = 1\} \text{ and } u \in U_{consis}\}$$  \hspace{1cm} (14)

$$e_v = \{e | (v_u, e) \in A \text{ and } v_u \in V_u\}$$  \hspace{1cm} (15)

When combined with the diffusion order, the above Eq.(15) can also be defined equally as follows:

$$e^k_v = \{t | (h, r, t) \in G \text{ and } h \in e^{k-1}_v\}, \hspace{1cm} k = 1, 2, \ldots, K$$  \hspace{1cm} (16)

where $k$ indicates the distance from the initial entity set. Given the definition of entity set, we then define the $k$-th order triple set for item $v$ as follows:

$$S^k_v = \{(h, r, t) | (h, r, t) \in G \text{ and } h \in e^{k-1}_v\}, \hspace{1cm} k = 1, 2, \ldots, K$$  \hspace{1cm} (17)

2) Knowledge-aware attentive: Given an entity set for consistent users interaction, we consider the role of the item’s inherent popularity while maintaining the connection between the head entity and the relation, which is formulated as follows:

$$z_1 = \text{LeakyReLU}\left(W_0\left(e^1_k\parallel e^1_v\right) + b_0\right)$$  \hspace{1cm} (18)

$$\pi\left(p_{hr}, e_v\right) = W^2_\pi \cdot \sigma\left(W_1 \cdot \text{pop}_\text{fun}\left(z_1, e_v\right) + b_1\right) + b_2$$  \hspace{1cm} (19)

where $z_1$ denotes the semantics representation of the head entity and relation aggregation. $\text{pop}_\text{fun}(\cdot)$ is a function, and the addition is found with the best performance in the experiments. Hereafter, we normalize the coefficients across the whole triples in the triple set by adopting the softmax function:

$$\pi\left(p^i_{hr}, e_v\right) = \frac{\exp\left(\pi\left(p^i_{hr}, e_v\right)\right)}{\sum_{(h', r', t') \in S^i_h} \exp\left(\pi\left(p^i_{hr}, e_v\right)\right)}$$  \hspace{1cm} (20)

where $\pi\left(p^i_{hr}, e_v\right)$ controls the attentive weight generated from aggregated semantics of head entity and relation and item’s inherent popularity. On the item side, we build the tail entity attentive embedding $b^i_{(v)}$ as follows:

$$b^i_{(v)} = \pi\left(p^i_{hr}, e_v\right) e^i_t$$  \hspace{1cm} (21)

Finally, we obtain a representation of the $k$-th order triple set for item:

$$e^k_{(v)} = \sum_{i=1}^{\left|S^k_v\right|} b^i_{(v)} \hspace{1cm} k = 1, 2, \ldots, K$$  \hspace{1cm} (22)

where $\left|S^k_v\right|$ is the number of triples in set $S^k_v$.

D. Model Prediction and Loss Function

After summarizing into $K$ sub-views and iterating the $K$-order aware process, we obtain the embedding set of user and item with $e^k_u$ and $e^k_{(v)}$ for $k = \{1, 2, \ldots, K\}$. And then for each user $u$, his final embedding is denoted as:

$$e^*_u = \left\|e^1_u \parallel e^2_u \parallel \ldots \parallel e^K_u\right\|$$

that concatenates his embedding at each sub-view. Similarly, each item $v$ final embedding is:

$$e^*_v = \left\|e^1_v \parallel e^2_v \parallel \ldots \parallel e^K_v\right\|$$

Finally, the predicted rating is represented by the inner product of the final user and item embeddings:

$$\hat{y}_{uv} = e^*_u \cdot e^*_v$$  \hspace{1cm} (23)

For each user, we extract the same number of negative samples as positive samples. Afterward, we have the following loss function for CRCC:

$$L = \sum_{(u, v, j) \in Y} -\ln \sigma\left(\hat{y}_{uv} - \hat{y}_{uj}\right) + \lambda\|\Theta\|^2$$  \hspace{1cm} (24)

where $Y = \{(u, v, j) | (u, v) \in Y^+, (u, j) \in Y^-\}$, $\sigma(x)$ is the sigmoid function, $\Theta$ is the model parameters set, and $\|\Theta\|^2$ is the $L2$-regularizer that parameterized by $\lambda$.

V. EXPERIMENT

A. Experiment Settings

1) Datasets and Evaluation Metrics: We adopt three benchmark datasets related to Microsoft KG Satori. The detailed statistics are summarized in Table I. We employ two widely used metrics $AUC$ and $F1$ to evaluate the performance of CTR prediction, and then choose Recall@K to evaluate the effectiveness of top-K recommendation. Note that higher values of the three metrics indicate better performance.

2) Baselines: We compare CRCC with the following baselines. BPRMF [18] is a CF-based method that optimizes implicit feedback using pairwise matrix factorization. CKE [13] combines the CF module with textual, structural, and visual knowledge embeddings in a unified framework. PER [19] is a path-based method that treats the KG as a heterogeneous information network and extracts meta-path based on features. RippleNet [10] propagates users’ potential preferences in the KG to enrich user representations. KGNN [15] is the first work to integrate GCN to KG-based recommendation, which iteratively aggregates information about neighboring entities. KGNN-LS [16] transforms a heterogeneous knowledge graph into a user-specific weighted graph. KGAT [17] uses an attention mechanism to prioritize neighbors in collaborative knowledge graphs. CKAN [11] propagates user interactive data layer by layer to high-order entities to learn user preferences and item embedding. KGIN [20] decouples user-item interactions at the granularity of user intents and implements GNN to the proposed user-intent-item-entity graph.

3) Parameter Settings: In CRCC, we divide each dataset into training, validation, and test sets with the proportion of 6:2:2. The embedding size is fixed at 64, the learning rate is set as 0.002, and the coefficient of $L2$ normalization is tuned as 0.00001. The size of the user’s triple set is set to 8 or 16, and then the size of the item’s triple set is fixed at 64. The batch size is fixed at 2048. We set the epoch on all three real-world datasets to 20, each experiment is repeated five times, and the average performance is reported.

B. Overall Comparison

Table II and Fig. 3 show that CRCC has obvious advantages over existing state-of-the-art baselines. Analyzing such performance comparison, we have the following observations:

- **Table II** and **Fig. 3** show that CRCC has obvious advantages over existing state-of-the-art baselines. Analyzing such performance comparison, we have the following observations:
Table I. Statistics of the three real-world datasets.

<table>
<thead>
<tr>
<th></th>
<th>Last.FM</th>
<th>Book-Crossing</th>
<th>MovieLens-1M</th>
</tr>
</thead>
<tbody>
<tr>
<td># users</td>
<td>1872</td>
<td>17800</td>
<td>6036</td>
</tr>
<tr>
<td># items</td>
<td>3846</td>
<td>14967</td>
<td>2445</td>
</tr>
<tr>
<td># interactions</td>
<td>42346</td>
<td>139749</td>
<td>753772</td>
</tr>
<tr>
<td># entities</td>
<td>9366</td>
<td>77903</td>
<td>182011</td>
</tr>
<tr>
<td># relations</td>
<td>60</td>
<td>25</td>
<td>12</td>
</tr>
<tr>
<td># triples</td>
<td>15115</td>
<td>151500</td>
<td>1241996</td>
</tr>
</tbody>
</table>

Table II. The result of AUC and F1 in CTR prediction.

<table>
<thead>
<tr>
<th>Model</th>
<th>Last.FM</th>
<th>Book-Crossing</th>
<th>MovieLens-1M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AUC</td>
<td>F1</td>
<td>AUC</td>
</tr>
<tr>
<td>BPRMF</td>
<td>0.756</td>
<td>0.701</td>
<td>0.658</td>
</tr>
<tr>
<td>PER</td>
<td>0.641</td>
<td>0.603</td>
<td>0.605</td>
</tr>
<tr>
<td>CKE</td>
<td>0.747</td>
<td>0.674</td>
<td>0.676</td>
</tr>
<tr>
<td>RippleNet</td>
<td>0.776</td>
<td>0.702</td>
<td>0.721</td>
</tr>
<tr>
<td>KGNN</td>
<td>0.796</td>
<td>0.721</td>
<td>0.684</td>
</tr>
<tr>
<td>KGNN-LS</td>
<td>0.805</td>
<td>0.722</td>
<td>0.676</td>
</tr>
<tr>
<td>KGAT</td>
<td>0.829</td>
<td>0.742</td>
<td>0.731</td>
</tr>
<tr>
<td>CKAN</td>
<td>0.842</td>
<td>0.760</td>
<td>0.754</td>
</tr>
<tr>
<td>KGIN</td>
<td>0.849</td>
<td>0.767</td>
<td>0.777</td>
</tr>
<tr>
<td>CRCC w/o K&amp;S</td>
<td>0.859</td>
<td>0.778</td>
<td>0.766</td>
</tr>
</tbody>
</table>

- CRCC consistently outperforms all baselines in all metrics across three datasets. Specifically, CRCC improves over the state-of-the-art baselines w.r.t. AUC by 1.2%, 1.7% and 1.3% in Last.FM, Book-Crossing and MovieLens-1M, respectively.
- Compared with CKAN and KGIN, the performance of CRCC justifies the effectiveness of the consistency of collaborative relation context. The results compared with KGNN and KGNN-LS show the significance of explicitly encoding collaborative signals.
- Comparing BPRMF with CKE, using KG significantly improves the performance of matrix factorization. This finding is also reflected in KTPU.
- The efficiency of a model is dependent on how it uses the KG information. The CF-based matrix factorization method BPRMF outperforms the path-based model PER, perhaps since it’s not always possible to find the optimal meta-path. The unified approach outperforms embedding-based and path-based baselines.

C. Ablation Study

To evaluate the efficacy of each component in our proposal, we compare CRCC with three variants and Table III shows the experimental results.

- CRCC w/o K&F : This variant is CRCC without the knowledge feature learning component.
- CRCC w/o R&K : This variant is CRCC without the collaborative relation context consistency component.
- CRCC w/o K&S : This variant is CRCC without the knowledge structure-aware component.

The following conclusions can be drawn from an analysis of the data in Table III: (1) CRCC w/o K&F can impair part of the performance, which demonstrates the utility of segmenting a user’s collaborative knowledge graph into a series of sub-views and learning related entity information separately. (2) The performance of recommendations is significantly reduced by CRCC w/o R&K, which emphasizes the importance of sampling high consistency score data between the user and his neighbors and their interacted items. (3) CRCC w/o K&S can substantially worsen recommendation performance, which is especially apparent on Last.FM and Book-Crossing. This demonstrates the benefits of considering the item’s inherent popularity while maintaining consistency in the diffusion entities with which users have interacted. Overall, it is clear that CRCC consistently achieves the best results.

D. Sensitivity Analysis

1) Impact of Dimension of Embedding: Table IV shows that increasing the embedding dimension improves the performance of CRCC within a certain range. But CRCC performance would decline with excessive dimension. This is due to the fact that when the embedding dimension increases, more information is encoded into it, but it also causes a little overfitting problem.

2) Impact of Numbers of Sub-view and Order: As shown in Table V, the best performance is achieved when k is 3, 3 and 2 in Last.FM, Book-Crossing and MovieLens-1M, respectively. This may be due to a trade-off between the number of segmented views and the order of knowledge diffusion. Too few segmented views and knowledge diffusion order are insufficient to capture entity relationships, but too large numbers may contain irrelevant noise information.

3) Impact of Size of Triple Set: Table VI demonstrates that the optimal size of the item triple set for MovieLens-
Recall@K

Fig. 3. The result of Recall@K in top-K recommendation.

1M is 64, which confirms that increasing the size can include more related entity sets and improve the recommendation performance. The best performance is achieved when the user triple set is set to 16. When it surpasses a reasonable threshold, the recommendation result decreases, thus we need a suitable size of user triple set.

VI. CONCLUSION

In this paper, we focus on the CRCC, which links the consistency score between the items that users and neighbors have interacted with as the fusion basis. This is the first work to address the inconsistency of collaborative relation context on KG-based recommendation. Extensive experiments on three real-world datasets demonstrate the effectiveness of CRCC. In the future, we will emphasize the effective utilization of multi-modal learning on collaborative knowledge graphs for better recommendation performance.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China (Nos.62262003 and U21A20474), the Guangxi Science and technology project (Guike AA22067070), the Guangxi Natural Science Foundation (No.2020GXNSFAA297075), the Center for Applied Mathematics of Guangxi (Guangxi Normal University), the Guangxi “Bagui Scholar” Teams for Innovation and Research Project, the Guangxi Collaborative Innovation Center of Multi-source Information Integration and intelligent Processing, the Guangxi Talent Highland project of Big Data Intelligence and Application.

REFERENCES

[19] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvasi Khandelwal, Brandon Norick, and Jiawei Han, “Personalized entity recommendation: A heterogeneous information network approach,” in WSDM, 2014, pp. 283–292.
An Effective Method for Constructing Knowledge Graph to Search Reusable ROS Nodes

Yuxin Zhao\textsuperscript{1,2}, Xinjun Mao\textsuperscript{1,2}, Sun Bo\textsuperscript{1,2}, Tanghaoran Zhang\textsuperscript{1,2}, Shuo Yang\textsuperscript{1,2}

\textsuperscript{1}College of Computer, National University of Defense Technology, Changsha, China
\textsuperscript{2}Key Laboratory of Software Engineering for Complex Systems, National University of Defense Technology, Changsha, China
\{yuxinzhaos, xjmao, sunbo, zhangthr, yangshuo11\}@nudt.edu.cn

Abstract—Developing robot software is difficult for most software engineers as it requires multi-discipline knowledge such as robotics, AI, and software engineering. Robot Operating Systems (ROS) provides a software development framework and lots of reusable ROS Nodes that encapsulate various robotics functions, which can simplify robot software development in terms of software reuse. However, searching and reusing required ROS Nodes from thousands of ROS Nodes is still challenging due to the scattered distribution of ROS Node information and the need for adequate search methods. In this paper, we present an effective method to construct a ROS Node knowledge graph in support of searching and reusing ROS Nodes. Our method uses multiple data sources, including open-source ROS software in Github and ROS wiki community. We extract two-tuple functional information and task-related noun phrases from the ROS Node description and ROS communication interactions from the ROS Node source code. The constructed ROS Node knowledge graph (RNKG) contains 14,965 entities and 15,767 relations. It provides rich semantic information to comprehensively and precisely describe ROS Nodes, their services, and related interaction topics and messages.

Index Terms—Knowledge Graph, ROS Nodes, Robotics software development

I. INTRODUCTION

Robot gradually plays an essential role in many fields, including autonomous vehicles, factories, healthcare, services, and commerce. A robot is a software-intensive system that needs several robotics software components to accomplish task requirements cooperatively. The advent of Robot Operating System (ROS)\textsuperscript{12} simplifies robot software development. ROS is the most popular framework in robotics which is designed to be modular at a fine-grained scale, and its fundamental concepts \textit{Node}, \textit{Message}, \textit{Topic}, and \textit{Service}. ROS Node can offer a specific function for robotics tasks and a sequence of ROS Nodes to complete complicated robot mission\textsuperscript{14} to construct robotics software. So, ROS Node is a proper component to reuse for developing robotics software. However, without in-depth knowledge of ROS Nodes and their interactions, reusing ROS Nodes is difficult\textsuperscript{1}. From a developer’s point of view, developing robotics software usually requires a series of ROS Nodes from different ROS packages. It is especially difficult for developers with less experience to find ROS Nodes and make ROS Nodes cooperative. Thus, in ROS-based robotics software development, a large portion of development efforts has been investigated into searching ROS Node, especially when their descriptions information and interactions are difficult to obtain. Therefore, searching applicable ROS Nodes is necessary for more efficient software development.

In this paper, we constructed a ROS Node Knowledge Graph (RNKG), a domain knowledge graph that captures the basic computation graph concepts of ROS and the relationship between different ROS concepts. The KG serves as the foundation for recognizing developer demands, inferring developer needs, recommending a proper ROS Node, and offering ROS knowledge for accelerating robotics software development. Our main contributions are summarized as follows:

- present a systematic method to construct a ROS domain knowledge graph, extracting two-tuple functional information and task-related noun phrases from the ROS Node description, and ROS communication interactions from the ROS Node source code.
- propose RNKG, a domain knowledge graph that provides rich semantics information to comprehensively and precisely describe the ROS Nodes, their services, and related interaction topics and messages.
- evaluate the completeness and the correctness of RNKG to show that the ROS knowledge extraction method is effective.

The rest of the paper is organized as follows. In the next section, we review previous studies on this topic. We illustrate the methodology of constructing a ROS Node knowledge graph in Section III. We evaluate the completeness and correctness of constructed knowledge graph in Section IV. We discuss the threats to validity in Section V. Finally, we conclude our work and discuss future directions in Section VI.

II. RELATED WORK

Due to the excellent performance of knowledge graphs in various application scenarios (question answering, recommendation, and information retrieval), the research on domain knowledge graphs\textsuperscript{7} in various fields continues to deepen. However, due to the strong domain characteristics of the collected knowledge, researchers need to design different knowledge extraction methods and relation extraction methods to solve the knowledge graph construction tasks in different knowledge types like extracting knowledge from network...
encyclopedia and web content [15], from Wikipedia [16] and Chinese texts [8].

Due to the powerful knowledge reasoning ability and recommendation of knowledge graphs, knowledge graphs have also been deeply studied in the field of robotics and have made a lot of breakthroughs. Applying the knowledge graph to the knowledge reasoning state enhances the robot’s ability to process environmental information [5]. Besides, it plays an indispensable role in promoting the development of robot software based on the power of recommendation. The ROS-related knowledge graph can be used in searching ROS Package [4] and ROS message [2] to simplify the process of developing robotics software.

In the research process, the knowledge reasoning ability and recommendation searchability of the knowledge graph will effectively help ROS developers find the ROS Node they need.

III. METHODOLOGY

This section introduces the ontologies of knowledge in our KG, and then we will describe our method to construct a robotics-specific knowledge graph RNKG. Fig[1] shows the whole method of our approach and introduces the method of constructing the ROS Node Knowledge Graph.

A. The ontology of RNKG

Before building the knowledge graph, we need to clarify the ontology and relationships contained in the ROS Node knowledge graph, so as to realize the guidance of entity extraction and relationship extraction. We show our core ontology in Fig[2] The ontology in the knowledge graph includes description information for ROS Node and common concepts of basic Computation Graph concepts of ROS.

B. Extract ROS Node Communication Information

We extract entity and relation related to ROS Node communication from the ROS Node source code in GitHub. To successfully extract the ROS Node information we need, we use pattern-matching based method [3] to realize the entity extraction of ROS Node.

C. Extract ROS Node Functional and Noun Features

We extract two-tuple functional information and task-related noun phrases from ROS Node description to distinguish different ROS Nodes.

Since ROS knowledge has strong domain characteristics, there are a large number of abbreviations (e.g., "ICP" means "Interactive Closest Point"). To fully understand the knowledge of ROS Node description, we build a local ROS dictionary and replace the ROS domain concepts in the text (specialized Nouns) with natural language description that users are more familiar.

For the ROS Node description, the type of each word tag is determined by the Part-Of-Speech Tagger in Fig[3] The part-of-speech tagger determines the syntactic category of each token and stores it in the POS feature, encoded in uppercase abbreviations. [2].

![Fig. 3. An example of Part-Of-Speech tagger.](image)

The next step in explaining ROS Node description is to eliminate the meaning of words. We need to combine single words into the most extensive phrase as much as possible to enhance the knowledge extraction of ROS Node description. We use two-tuple functional information and task-related Noun phrases to summarize ROS Node description.

For two-tuple functional information, we rely on the term Dependency Parsing (DP) [11] to examine the dependencies between the phrases of ROS Node description in Fig[4] We use 'obj': (register, pointclouds) to describe the function of ROS Node.

![Fig. 4. An example of extracting two-tuple functional information.](image)

For task-related Noun phrases, we use NP-chunking [17] to extract the ROS Node-related information in Fig[5] and we can get task-related Noun phrases "two pointclouds, interactive closest point, registration technique".

D. Constructed RNKG

So far, we have successfully constructed a knowledge graph containing ROS Node knowledge using the previous three extraction knowledge methods. The resulting RNKG consists of 14,065 entities and 15,767 relationships. Using Neo4j [10]
to form the ROS Node domain knowledge graph, we store the structured triad data.

IV. EVALUATION

In our constructed RNKG, the quality of ROS Node knowledge is essential, so we measure the completeness and correctness of ROS Node knowledge extraction to verify the usefulness of the ROS Node knowledge extraction method.

- **RQ1**- How is the completeness of the knowledge captured in the constructed RNKG?
- **RQ2**- How is the correctness of the knowledge captured in the constructed RNKG?

A. The completeness of RNKG.

The dataset required to build the knowledge graph was collected in GitHub. The completeness of RNKG refers to whether it contains all needed ROS Node information, including ROS Node Name, Topic, Service, and Message.

1) **Protocol**: we measure whether RNKG contains all ROS Node-related knowledge. Based on the relevant conclusions of other studies [6] [13], we constructed a verification dataset (randomly extracting 345 Node data, 345 Topic data, 345 Service data, and 345 message data) from ROS Wiki. The measure is whether the extracted validation data can be found in RNKG.

2) **Results and Analysis**: The results are shown in Table I. 90% of the ROS Node, Topic, and Service in the verification dataset can be found in RNKG, and 98% of ROS Node information in the verification dataset can be found in RNKG. Although the Message in the verification dataset is less than 90%, its accuracy is still considerable. These data fully illustrate that the ROS Node knowledge contained in RNKG is complete.

![More than 90% ROS Node knowledge extracted from ROS Wiki can be found in RNKG.](image)

B. The correctness of Node knowledge in RNKG

To evaluate the correctness of entity extraction method and relation extraction method, we should check whether we proposed method can successfully extract the required entities (sub_topics, pub_topics, service, service_message, package, and repository), and build correct relation between different entities.

1) **Protocol**: We obtained the knowledge of 355 ROS Nodes from ROS Wiki and RNKG, respectively. The knowledge provided in ROS Wiki is correct, and we need to measure the correctness of the knowledge in RNKG. We will calculate the correctness of entity extraction and relation extraction separately.

2) **Results**: The results of entity extraction are shown in Table II Generally speaking, the entity extraction of the RNKG knowledge graph is accurate and as complete as possible. The main reason for the high accuracy of entity extraction is that RNKG is in the ROS domain, and the dataset is composed of source code. Entity extraction from code data is simpler than analyzing language from natural language. The format of code data is fixed and strict, so it will naturally have a higher accuracy rate when combined with a template-based approach.

From Table II shows that the $F_1$ of sub_topics and pub_topics is 87.60%, which is relatively low compared to other entities. One of the main reasons for this is that the code parameter settings of different ROS Node programs are different, and entity extraction is very difficult. It is difficult to distinguish what kind of parameter it is according to the string, so the accuracy of sub_topics and pub_topics is not very high.

For the relationship extraction between different entities, we determine it by analyzing the unique API in the ROS program because different entities must be implemented through specific API calls, so for the relation-

### TABLE I
**THE COMPLETENESS OF RNKG.**

<table>
<thead>
<tr>
<th>Category</th>
<th>Node name</th>
<th>Topic</th>
<th>Service</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>From ROS Wiki</td>
<td>345</td>
<td>345</td>
<td>345</td>
<td>345</td>
</tr>
<tr>
<td>From RNKG</td>
<td>340</td>
<td>332</td>
<td>317</td>
<td>308</td>
</tr>
<tr>
<td>Contained Percentage</td>
<td>98.55%</td>
<td>96.23%</td>
<td>91.88%</td>
<td>89.27%</td>
</tr>
</tbody>
</table>

### TABLE II
**EXPERIMENTAL RESULTS OF ENTITY EXTRACTION.**

<table>
<thead>
<tr>
<th>Category</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub_topics</td>
<td>87.50%</td>
<td>87.72%</td>
<td>87.60%</td>
</tr>
<tr>
<td>pub_topics</td>
<td>87.93%</td>
<td>86.44%</td>
<td>87.17%</td>
</tr>
<tr>
<td>service</td>
<td>95.71%</td>
<td>97.14%</td>
<td>96.14%</td>
</tr>
<tr>
<td>service_called</td>
<td>89.71%</td>
<td>86.97%</td>
<td>88.31%</td>
</tr>
<tr>
<td>service_message</td>
<td>93.63%</td>
<td>88.55%</td>
<td>91.01%</td>
</tr>
<tr>
<td>topic_message</td>
<td>92.07%</td>
<td>94.37%</td>
<td>93.20%</td>
</tr>
<tr>
<td>package</td>
<td>93.45%</td>
<td>91.74%</td>
<td>92.58%</td>
</tr>
<tr>
<td>repository</td>
<td>94.31%</td>
<td>96.51%</td>
<td>95.38%</td>
</tr>
</tbody>
</table>

### TABLE III
**EXPERIMENTAL RESULTS OF RELATION EXTRACTION.**

<table>
<thead>
<tr>
<th>Category</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(sub_topics, message_type)</td>
<td>69.55%</td>
<td>81.72%</td>
<td>75.14%</td>
</tr>
<tr>
<td>(pub_topics, message_type)</td>
<td>87.72%</td>
<td>86.44%</td>
<td>87.07%</td>
</tr>
<tr>
<td>(service, message_type)</td>
<td>83.80%</td>
<td>87.14%</td>
<td>85.43%</td>
</tr>
<tr>
<td>(service_called, message_type)</td>
<td>74.52%</td>
<td>86.97%</td>
<td>80.26%</td>
</tr>
</tbody>
</table>
ship in RNKG ("has_subscribed_topics, has_published_topics, has_service, has_service Called, has_package, has_repos, has_features") are accurate. But we need to specifically explain "has_message_type" in Table III to explain some potential problems and impacts we found in verifying the correctness of the relationship. We can see that the $F_1$ of (sub_topics, message_type) and (service, message_type) are 87.07% and 85.43% respectively, which are relatively high. But the $F_1$ of (pub_topics, message_type) and (service_called, message_type) are 75.14% and 80.26% respectively, which are relatively low. The main reason is that when extracting subscribe topics and service called relationships, we need to analyze the Callback Function further. This process will cause errors in identification information and interfere with relationship extraction.

The ROS Node knowledge extracted by our method has more than 80% correctness and can be used to guide practice and development.

V. Threats to Validity

External validity. For the ROS-related extraction work of the dataset, we admit that our entity extraction method and relation extraction method from ROS Code has some flaws. There is still a large amount of knowledge in the code, which requires more detailed mining. Although there may be some erroneous data in the knowledge graph dataset and missing information in data extraction, we do not consider this a significant threat to external validity.

Internal validity. In the experimental verification process of RNKG, we randomly verified the completeness and correctness of ROS Node in RNKG. Additionally, we randomly selected ROS Node to evaluate the effectiveness of RNKG.

VI. Conclusion and Future Work

In this paper, we propose a method to construct a robotics knowledge graph RNKG with 14,065 entities and 15,767 relations for better serving ROS developers. We systematically introduce how it is automatically built from source code knowledge from GitHub and text description from ROS Wiki. We proposed an algorithm for searching ROS Node based on user description. Finally, we use three experiments to illustrate the information quality of RNKG and the possibility of implementing ROS Node search.

In the future, we plan to do in-depth follow-up research based on RNKG. And after in-depth mining of robot software project, the new data acquired by it is integrated into the constructed RNKG. We will continually enlarge RNKG to cover emerging ROS Node. On the other hand, we can better apply the reasoning ability of the knowledge map to the development tasks of robot software and use better semantic technology to support developers in searching for the ROS Node they need.

VII. Acknowledgement

This work was supported by the Key Laboratory of Software Engineering for Complex Systems and the National Science Foundation of China under granted number 62172426.

REFERENCES


Multi-Domain Feature Representation and Multi-Dimensional Feature Interaction for Person-Job Fit

Bin Li, Jianwen Ding, Zhennan Chen, Hanpeng Jiang, Fan Lin*
School of Informatics, Xiamen University, Xiamen, China
{30920211154159, 30920201153956, znchen, 30920221154272}@stu.xmu.edu.cn, *iamafan@xmu.edu.cn

Abstract—Person-job fit aims to use the algorithms to match jobseekers with job postings to overcome information overload on online recruitment platforms. Traditional matching algorithms are not ideal in the feature representation and interaction of resumes and job postings. To this end, we propose a person-job fit model PJFFRFI based on multi-domain feature representation and multi-dimensional feature interaction, which comprehensively considers the features of various domains and learns feature correlation vectors in different dimensions. Specifically, we first divide the features in resumes and job postings into seven domains, and design different representation methods according to the data type. Then we propose a feature enhancement module (FEM) based on multi-head self-attention to learn the feature correlation vectors in resumes and job postings. Moreover, we propose a feature interaction module (FIM) to facilitate feature interaction both inside and outside the domain. Extensive experiments on a real-world dataset demonstrate that the proposed method significantly surpasses the state-of-the-art methods.

Index Terms—Feature representation, Multi-dimensional feature interaction, Feature enhancement, Person-job fit

I. INTRODUCTION

Due to the epidemic, online recruitment has recently taken over as the primary method [1] in the job market, making it simpler for both jobseekers and employers to find the right jobs and qualified candidates. By the end of 2022, LinkedIn had over 850 million users and 58 million companies registered in more than 200 countries. Due to the huge amount of data and fast-paced requirements of the job market, it is difficult to meet these requirements only through manual review. Therefore, it is extremely urgent to design an effective person-job fit algorithm to quantify the matching degree between jobseekers and job postings.

In the recent job market, a typical job matching idea is to match the requirements released by the employer with descriptive data such as experience of the jobseeker in Fig. 1. Thus, a series of deep learning algorithms dedicated to solving text matching came into being. Zhu et al. [2] proposed the PJFNN based on CNN, which adopted the hierarchical representation structure that can identify specific requirements the candidates meet in the job posting. Qin et al. [3] proposed a RNN model based on a hierarchical attention to learn word-level semantic representations of resumes and jobs and ability perception representations with different levels.

However, these models only focus on text features, ignoring the possible impact of structured data (The blue part of Fig. 1). For example, when the job city posted by the employer is not in the jobseeker’s plan, the jobseeker has a high probability of rejecting the job even if the text matching score is high. Furthermore, [4] introduced click stream and browse volume as structured data to assist text data, [5]–[7] also introduced different features such as numerical features identified as structured data, and considered the possible impact of interaction between structured data and textual data on matching results. Nevertheless, they did not fully mine the implicit feature interaction information from multiple dimensions. On the one hand, there is often a strong correlation between structural features. For instance, the more developed the city, the higher the salary level. Similarly, salary has a positive correlation with education and working years. On the other hand, there are also plentiful hidden information within the same domain features interaction and between different domain features interaction from resumes and job postings.

To sum up, we propose a person-job fit model based on multi-domain feature representation and multi-dimensional feature interaction, termed PJFFRFI. Our proposed model can capture the comprehensive interaction features based on learning different types of feature representations given a job posting and a resume. Specifically, we divide all the features in
resumes and job postings into three types as shown in Table 1, and adopt different embedding methods according to the data type. Meanwhile, considering the correlation between structure features, a feature enhancement module (FEM) is introduced to learn the feature correlation both in resume and job postings. Along this line, we propose a feature interaction module (FIM) to extract the hidden correlation features between resumes and job postings in the same domain and the hidden feature combinations between domains.

Our contributions can be summarized as follows:

- We propose a novel person-job fit model named PJFFRFI by learning embedding vectors of different domains and extracting feature correlations from different dimensions.
- We propose a feature enhancement module (FEM) to introduce correlation features among various domains in job postings. Furthermore, we design a novel feature interaction module (FIM), which leverages inner and outer feature interactions to learn the implicit feature correlations.
- Extensive experiments demonstrate our superior performance over the recent state-of-the-art methods by a large margin.

II. RELATED WORK

In earlier study, person-job fit is regarded as a recommendation problem, relying on Boolean keyword matching which usually fails to give satisfactory recommendations. Malinowski et al. [8] proposed a bilateral selection model for the first time, which took the preference of both jobs and jobseekers into account and improved the recommendation performance. After that, the extensive application of collaborative filtering algorithm (CF) [9], [10] in job recommendation tasks has further promoted the progress of person-post matching tasks. However, the CF algorithm is based on historical interaction data and has the defect of cold start, which has attracted many experts to use different methods such as hybrid recommendation [11]–[13] to solve this problem.

Due to the advanced performance of deep learning technology in semantic mining, convolutional neural network (CNN) [2], recurrent neural network (RNN) [3], attention [14] and etc have been widely used in person-job fit. Luo et al. [15] integrated different types of information in a hierarchical representation and introduced adversarial learning to model job and resume representations. So as to make up for the semantic difference between job postings and resumes, Yao et al. [16] designed a knowledge-aware graph encoder and incorporated prior knowledge into graph representation learning to improve the performance.

In order to further enhance the semantic representation and learn more effective information in resumes and job postings, Bian et al. [4] proposed a multi-view co-teaching network from sparse interaction data, which introduced a relation-based module to complement the text-based matching module, realized the enhancement of semantic representation and data enhancement. FINN [5] divided features into three fields, and learned interaction signals of categorical features and textual features respectively. Jiang et al. [6] proposed a feature fusion method, which fused the expressive features of the job and candidate, so as to obtain a more comprehensive and effective representation. He et al. [7] proposed an end-to-end person-job fit model MUFFIN, which designed a module to learn the latent correlations between features in each field and a module based on multi-head self-attention with a residual connection to learn interactions.

Inspired by existing work, we propose a deep learning model to predict the matching scores between resumes and job postings. Compared with existing methods, our model divides all features into different domains and learns feature representations respectively. Moreover, we propose several feature interaction modules to extract feature correlations from multiple dimensions including within the job postings, within the domain, and between domains.

III. METHODOLOGY

A. Problem Definition

We denote the resume set as \( R = \{r_1, r_2, \ldots, r_m\} \) and job set as \( J = \{j_1, j_2, \ldots, j_n\} \), where \( m \) and \( n \) are total number of resumes and job postings respectively. Resume \( r_p \) and job posting \( j_q \) both have \( k \) domains, denoted as \( r_p = \{r_{p,1}, r_{p,2}, \ldots, r_{p,k}\} \) and \( j_q = \{j_{q,1}, j_{q,2}, \ldots, j_{q,k}\} \). Particularly, according to the characteristics of private dataset, \( r_{p,i} \) may consists \( m_i \) features, denoted as \( r_{p,i} = \{r_{p,i,1}, r_{p,i,2}, \ldots, r_{p,i,m_i}\} \) while \( j_q \) has only one feature in one domain. For example, current salary and desired salary listed in the resume belongs to one domain named salary. The recruitment records are denoted as the set of \( PJF = \{r_p,j_q,y_{p,q}\} \), where \( y_{p,q} \in \{0,1\} \). \( y_{p,q} = 1 \) means that the resume \( r_p \) successfully fits the job \( j_q \) while \( y_{p,q} = 0 \) means that the match fails. Our target is to design a deep learning model to predict \( y_{p,q} \).

B. Overview

As shown in Fig. 2, the proposed model PJFFRFI groups all features from resumes and job postings into \( k \) domains as input. In the embedding module, there are two methods mapping numerical domain features (e.g., Salary), categorical domain features (e.g., City) and textual domain features (e.g., Requirements and Experience) into two hidden spaces, and get the embedding vectors of different domains respectively, denoted as \( domain_i \) vector. Followed by a feature enhancement module (FEM), multi-head self-attention (MHSA) is used to introduce hidden correlation vectors between structured features, and output the feature enhancement vector in different domains, marked as \( domain_i \) _e_vector. Then, these vectors will be fed into the feature interaction module (FIM), including inner and outer interaction modules. Finally, we predict the matching score by prediction module.

C. Embedding Module

To begin with, we normalize numerical features with standard distributions and project categorical features into one-hot
vectors. To concatenate all types of features, each job and resume is represented as a vector. It is defined as:

\[
Input^R = [x^R_{1,1}, x^R_{1,2}, \ldots, x^R_{1,m_1}, \ldots, x^R_{k,m_k}],
\]

\[
Input^J = [x^J_1, x^J_2, \ldots, x^J_I],
\]

Where \(m_k\) represents the number of features of the \(k\)th domain in resume. \(x^R_{i,t}\) and \(x^J_i\) represent the input of the \(i\)th feature in the \(t\)th domain in the resume and the \(i\)th feature in the job posting respectively. When the \(i\)th domain is of numeric type, the \(x^R_{i,t}\) and \(x^J_i\) are scalar values. Otherwise, they are vectors.

Next, the numerical features and categorical features are mapped to a low-dimensional vector, which is defined as:

\[
emb^R_{i,t} = V_i x^R_{i,t}, emb^J_i = V_i x^J_i,
\]

where \(V_i\) is a shared embedding matrix. Meanwhile, textual features are embedded by a pretrained ALBERT [17]. The integration process can be denoted as:

\[
emb^R_{i,t} = \text{ALBERT}(x^R_{i,t}), emb^J_i = \text{ALBERT}(x^J_i)
\]

D. Feature Enhancement Module (FEM)

Since the basic information (education, living city etc.) of non-text features in resumes is more likely to be affected by factors such as the candidates’ family and social relations, reflects personalized job seeking tendency, therefore, it is of significance to study the relevance between non-text features in resumes. In the same way, analysis of structural features in job postings is helpful in understanding recruiters’ intentions, so we treat the embedding vectors of non-textual features as input and propose a FEM based on multi-head self-attention to introduce strong feature correlations both in resumes and job postings.

Multi-head self-attention has been successfully applied to semantic understanding [18], machine translation [19], etc. We apply it to capture the correlation between non-textual features in job postings and use the three most relevant features to achieve feature enhancement. The core of the self-attention is Query, Key, and Value. The multi-head self-attention is a combination of multiple self-attention modules, providing multiple representation subspaces to the attention layer. Taking the \(i\)th non-textual feature as an example, we calculate its similarity to other features and normalize with softmax. The correlation weight coefficient is calculated as:

\[
\alpha^g_{i,j} = \frac{(Q^gemb^J_i) \cdot (K^gemb^J_j)^\text{transpose}}{\sqrt{d}},
\]

\[
\tilde{\alpha}^g_{i,j} = \frac{\exp(\alpha^g_{i,j})}{\sum_j \exp(\alpha^g_{i,j})},
\]

\[
\alpha_{i,j} = \frac{1}{|G|} \sum_{g=1}^{|G|} \tilde{\alpha}^g_{i,j},
\]

where \(Q^g\) and \(K^g\) are the Query and Key of the \(g\)th head. \(d\) represents the dimension of \(emb^J_i\) and \(|G|\) represents the number of heads. We select the three most relevant features, multiply their embedding vectors by the corresponding weight coefficients and add them to the embedding vector of the \(i\)th feature to achieve feature enhancement. The final representation of the \(i\)th non-textual feature in the job posting can be defined as:

\[
emb^J_i = emb^J_i + \sum_{j \in \text{Top 3 related features}} \alpha_{i,j} \cdot emb^J_j
\]

E. Feature Interaction Module (FIM)

1) Inner Interaction Module: Different from the common method of using average pooling to compress multiple features in each domain in the resume, we interact each of the features in the resume with the feature in the job posting in the corresponding domain, calculate the difference and the Hadamard product between two embedding vectors respectively, introduce the distance correlation vector and angle correlation vector between the features and then concatenate them with the
two original embedding vectors to represent the interactions between resumes and job postings in the domain. Finally, the interaction vectors in the same domain are compressed by average pooling. The specific formula is as follows:

\[
inn_i = \frac{1}{m_i} \sum_{t=0}^{m_i} \{W_i[emb_{i,t}^R] \odot emb_{i,t}^l + emb_{i,t}^l - emb_{i,t}^l \oplus emb_{i,t}^R \odot emb_{i,t}^l + b_{i,t}\},
\]

where \( \oplus \) represents concatenation and \( \odot \) represents the Hadamard product. \( m_i \) denotes the number of features contained in the resume in the \( i \)th domain. \( W_i \) and \( b_{i,t} \) represent the parameters. In order to project all the domain features into the same dimensional space, we use a MLP layer to map all the vectors.

2) Outer Interaction Module: One of the core problems in feature interaction is to extract the hidden feature combinations. Hence, in outer interaction module, we model the domain feature vectors with the same dimension to learn the meaningful combinations of domains. We propose a Gated Linear Unit (GLU) [20] to capture feature interactions between domains. First, we concatenate the interaction vectors between resumes and job postings in each domain:

\[
Inn = inn_1 \oplus inn_2 \ldots \oplus inn_k,
\]

where \( \oplus \) represents concatenation and \( k \) represents the number of domains. \( Inn \) is fed to two convolutional layers in GLU. The output of the first convolutional layer with sigmoid function can help the model to learn the importance of features by backpropagation. The second convolutional layer has no activation function. The output of GLU is the Hadamard product of the output of the two convolutional layers, the formula is:

\[
\text{Conv}_1 = \frac{1}{1 + \exp[-(W_1 \ast Inn + b_1)]},
\]

\[
\text{Conv}_2 = W_2 \ast Inn + b_2,
\]

\[
\widehat{Out} = \text{Conv}_1 \odot \text{Conv}_2,
\]

where \( W_1, b_1, W_2, b_2 \) are parameters, \( \odot \) represents the Hadamard product. In order to avoid network degradation, we add a residual connection to reserve the interaction features between resumes and job postings learned by the inner interaction module. At last, the output of outer interaction module is:

\[
Out = \widehat{Out} + Inn
\]

F. Prediction Module

In prediction module, MLP fed with the hidden vectors of interactions between jobs and resumes and the combinations of interaction features between domains is introduced to predict the matching scores. To simplify, we use \( d^l \) to denote the input at 0th layer. Next, the output of each layer in the MLP can be formulated as:

\[
d^{l+1} = ReLU \left( W_{l+1}d^l + b_{l+1} \right),
\]

where \( d^{l+1} \) is the output of the \( l \)th layer, \( W_{l+1} \) and \( b_{l+1} \) are parameters. Then the predictive matching scores can be obtained through a two-dimensional vector which is the output of the last layer and are mapped into the matching probability through the softmax function:

\[
y_{p,q} = \text{softmax} \left( W_Ld^l + b_L \right),
\]

where \( L \) indicates the depth of the MLP.

Finally, we use the cross-entropy loss function to optimize:

\[
\text{Loss} = -\frac{1}{N} \sum \left[ y_{p,q} \log \hat{y}_{p,q} + (1 - y_{p,q}) \log (1 - \hat{y}_{p,q}) \right]
\]

where \( y_{p,q} \) and \( \hat{y}_{p,q} \) represent the matching label and predictive matching score of \( < r_p, j_q > \) respectively. \( N \) indicates the total number of training samples.

IV. EXPERIMENTS

A. Experimental Setup

1) Dataset: Our experiments use private data provided by recruitment platforms and companies. Overall, our dataset consists of 35,413 job postings, 1,648 resumes and 43,504 recruitment records. To meet the experimental conditions, we removed the job posting with null requirement and resume with null experience, and replaced the null value in the structured feature with "other". And we filter the dataset to make each resume have more than 6 suitable jobs and at least 3 unsuitable jobs to simulate the actual recruitment market. Finally, we get 43,504 recruitment records, among which 25,032 are positive and 18,472 are negative. Fig. 3 shows the distribution of person-job fit. It can be seen from (a) and (b) that most jobs only recruit one person, some recruit 2 to 4 people, and a few recruit more than 4 people. (c) shows that applicants generally send their resumes to multiple companies to increase employment opportunities, while (d) explains that a resume will usually receive offers from 4 to 21 companies. Through statistics, we found that almost all work experience resumes are less than 300 words, and most of the job descriptions are less than 500 words. In addition, Table 1 shows the 7 domains and 3 categories we designed.

2) Hyper-parameter Setting: In our experiments, we set the max number of words as 300 and 512 for experience of each resume and requirements of job posting respectively according to the statistical results. The dimension of numerical and categorical embedding vectors is 32 while the output dimension of ALBERT is 312. FEM uses a two-headed self-attention network with an embedding size of 32. For GLU in

| TABLE I |
|Domains| Type| Features in resumes| Features in job postings|
|Working years| Numerical| Working years| Minimum working years|
|Salary| Numerical| Desired salary| Current salary| Minimum Salary|
|Education| Categorical| Academic qualifications| Academic requirements|
|City| Categorical| Living city, Top 3 desired cities| Working location|
|Industry| Textual| Desired industries| Job industry|
|Job type| Textual| Current job title| Job title|
|Description| Textual| Top 3 desired job titles| Requirements|

| Working years| Salary| Education| City| Industry| Job type| Description| Total|
| 0-5 years| $30,000-$40,000| Bachelor| City 1| Industry 1| Job type 1| Description 1| 20|
| 6-10 years| $40,000-$50,000| Master| City 2| Industry 2| Job type 2| Description 2| 20|
| 11+ years| $50,000-$60,000| PhD| City 3| Industry 3| Job type 3| Description 3| 20|

336
the outer interaction module, we set the size of embedding vectors to 32. The list of dimensions for the hidden layers in the MLP is $[224, 64, 32]$. Additionally, we set the batch size to 16 and use an early stop to prevent overfitting with a patience of 30 epochs.

3) Baselines: In order to verify the effectiveness of our proposed model PJFFRFI, we selected the following baselines, including 3 models with input features as text (t) and 6 models with input features as text and structure (s + t).

- PJFNN [2] uses a bipartite neural network to learn a joint representation of person-job fit from historical job applications, thereby mapping job posting and resume features to a shared latent representation.
- APJFNN [3] proposes a word-level semantic representation for both job requirements and jobseekers' experience based on RNN to exploit the rich information available at abundant historical job application data. BPJFNN could be treated as a simplified version of APJFNN model.
- Random Forests (RF), Gaussian Naive Bayes (GaussianNB) and Decision Trees (DT). We feed the embedding vectors for each resume and job posting from Section 3.3 as input to these models.
- MUFFIN [7] groups all the features into several fields, and proposes two modules to learn the latent correlations in each fields and the field interactions.

4) Evaluation Metrics: We quantify the performance of each model using evaluation metrics commonly used in classification and recommendation tasks: accuracy, AUC, F1 score, precision, mean average precision (MAP).

B. Experimental Results

The overall performance of the models is shown in Table 2, where the best results in terms of different evaluation metrics are marked in bold. Compared with other models, our proposed model PJFFRFI achieves the state-of-the-art results in all evaluation metrics. It indicates that our proposed feature interaction in three dimensions improves the performance of predicting matching scores between resumes and job postings. First, compared to models that only consider pure text features, the performance of models whose input features are text and structure features is generally higher than that of models with pure text features. Taking the best-performing APJFNN as an example, our proposed model improves 8.61%, 5.36%, 6.49%, 7.61%, 28.52% in terms of accuracy, AUC, F1 score, precision, MAP respectively, indicating that introduce of structure data does help to improve the performance of the model. Secondly, in models that considers structured data and text data, deep learning models far exceeds machine learning models such as DT, RF, Gaussian NB in all indicators, shows that deep learning is effective in semantic mining and structured data representation. Finally, compared with the models MUFFIN, PJFF and MV-CoN whose model structure is similar to ours, our performance has also improved significantly. Compared to the muffin with the best overall performance in baselines, PJFFRFI improves 5.07%, 6.44%, 7.49%, 4.14%, 5.71% in five metrics, which explains that the model we propose is better at semantic representation and mining the implicit relationship in interaction features.

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy</th>
<th>AUC</th>
<th>F1 Score</th>
<th>Precision</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJFFRFI (no_structure)</td>
<td>0.6546</td>
<td>0.6363</td>
<td>0.4179</td>
<td>0.6796</td>
<td>0.653</td>
</tr>
<tr>
<td>PJFFRFI (no_sem)</td>
<td>0.6673</td>
<td>0.6852</td>
<td>0.5169</td>
<td>0.6983</td>
<td>0.675</td>
</tr>
<tr>
<td>PJFFRFI (no_inner)</td>
<td>0.6707</td>
<td>0.6961</td>
<td>0.5303</td>
<td>0.7249</td>
<td>0.681</td>
</tr>
<tr>
<td>PJFFRFI (no_outer)</td>
<td>0.6562</td>
<td>0.6900</td>
<td>0.4517</td>
<td>0.6991</td>
<td>0.677</td>
</tr>
<tr>
<td>PJFFRFI (no_residual)</td>
<td>0.6614</td>
<td>0.6852</td>
<td>0.5461</td>
<td>0.7025</td>
<td>0.671</td>
</tr>
</tbody>
</table>

C. Ablation Study

In this section, ablation experiments are conducted to study the contributions of some modules in PJFFRFI. Specially, we are interested in whether the structured data input and multi-dimensional feature interactions work and whether the residual connection network works. Therefore, we compare PJFFRFI with: 1) PJFFRFI without structured data input; 2) PJFFRFI without feature enhance module; 3) PJFFRFI without...
inner interaction module; 4) PJFFRFI without outer interaction module; 5) PJFFRFI without residual connection.

The results are shown in Table 3. It is apparent that removing the structured data input reduces the performance, indicating that structured data plays an important role in person-job fit. After removing PJFFRFI (no_fem), the overall performance of the model degrades 5.89%, 6.74%, 17.16%, 7.73%, 3.94% in five metrics respectively, which shows that FEM effectively identifies the implicit preference intention of recruiters and enhances it. We also observe that removing FIM, performance decrease by 5.35%-7.68%, 5.07%-6%, 14.2%-34.07%, 3.78%-7.61%, 2.95%-5.37% respectively, verifying the effectiveness of extracting hidden features within and between domains. The comparison of results between PJFFRFI (no_residual) and PJFFRFI confirms that residual connection makes positive contributions.

D. Case Study

We further demonstrate the effectiveness of our model by case studies. Fig. 1 shows two candidate jobs for the same resume. The matching scores predicted for Job posting 1 and Job posting 2 are 0.9106 and 0.1654 respectively. It is shown that our model is able to predict the matching scores precisely. Referring to textual features marked by gray boxes, the bold matching words between Job posting 2 and Resume are more than Job posting 1. However, we can observe that in structured domains, Job posting 2 cannot meet the city expectations of the jobseeker and has ambiguous educational requirements, leading to its rejection. To sum up, our model is capable of adequately simulating the behavior of human resource managers, focusing not only on text matching but also on the interactions of informative structured features (e.g., Salary and City).

V. CONCLUSION

In this paper, we propose a novel model named PJFFRFI based on multi-domain feature representation and multi-dimensional feature interaction for person-job fit. PJFFRFI divides all features into multiple domains, maps numerical and categorical features to a low-dimensional space and learns semantic representation using pretrained ALBERT. Then, we design a feature enhancement module (FEM) and a feature interaction module (FIM) with inner and outer interactions to model the feature interactions in multiple dimensions, including the feature correlations in job postings, the feature interactions between resumes and job postings in the same domain and the hidden feature combinations between domains. Finally, we evaluate our model using a real recruitment dataset. Experimental studies confirm the superiority of PJFFRFI over existing models and verify the contributions of each module in PJFFRFI.

REFERENCES

Abstract—Stack Overflow is a widely-used community Q&A website for programming-related queries. In such a platform, providing related questions as suggestions to the users can significantly enhance their search experience. Although there are many approaches based on deep learning that can automatically predict the relatedness between questions, those approaches are limited because the semantic and interaction features of the sentences may be lost. In this paper, we propose a novel method to predict the relatedness between questions based on semantic matching. We adopt the Interaction Feature Extractor to capture the interaction information and fuse it through a fusion mechanism to enhance the interaction between questions. Our experimental results demonstrate that our proposed method achieves state-of-the-art performance in terms of Precision, Recall, and F1-score evaluation metrics, outperforming the baseline approaches. Furthermore, we show that our model also performs well in other semantic matching tasks in software fields, indicating its generalization ability and robustness.

Index Terms—Stack Overflow, Question Relatedness, Deep Learning, Semantic Matching

I. INTRODUCTION

Stack Overflow is a Programming Community-based Question Answering (PCQA) forum that functions as a platform for developers to seek solutions to programming issues and exchange knowledge in their respective fields. Over time, Stack Overflow has accumulated a large number of questions, with many duplicate and related questions. Figure 1 illustrates an example of a pair of related questions. They are discussing different but related issues, which means that the answers in question 2 contribute to the solution of question 1. It is worth mentioning that typically, developers manually share related questions through URL links, which are identified after the question has been posted. If developers can find related questions when seeking solutions to new issues, they can leverage the existing answers to efficiently resolve their problems. Meanwhile, these available answers enable developers to avoid posting duplicate issues, thereby facilitating the maintenance of the website. The website contains a vast array of questions, and given the possibility of expressing the same question in multiple ways, manual identification methods prove to be inefficient and time-consuming. Hence, it is necessary to propose an automated method to identify duplicate and related questions.

Due to its strong nonlinear fitting ability, deep learning can effectively extract semantic information, and in recent work, some researchers have employed deep learning to predict question relatedness [1, 3, 6, 9]. Taking Pei et al.’s [3] work as an example, their method adopts BiLSTM to extract contextual information and employs the soft attention mechanism to extract the interaction features of words. However, their model only learns the interaction features of words and ignores the extraction of semantic and interaction features of sentences, which are crucial for predicting the semantic relevance between questions.

This paper conducts research based on the work of Pei et al. [3]. In order to leverage interaction features to predict the semantic relevance between questions, we propose a related questions Detection Model in Stack Overflow (named DMSO), which detects the related questions based on the semantic
relatedness between questions. To better extract the features of sentences, we introduce Interaction Feature Extractor to enhance the model’s ability to extract interaction information and fuse it through a fusion mechanism to enhance the interaction between questions. To evaluate the proposed model, we conduct experiments on the public dataset built by Shirani et al. [6] and compare with the baseline models.

The main contributions of this paper are as follows:

- We proposed a novel semantic matching model to detect the semantic relatedness between questions, which is an automated method to identify duplicate and related questions.
- We implement an interaction feature extractor to capture the features of sentences to enhance the model’s ability to extract interaction formation.
- We evaluate our approach on a public dataset, and the result shows that our model not only outperforms previous methods in the question relatedness prediction task but also has a good performance in duplicate question detection tasks in software engineering domains.

The rest of the paper is organized as follows. Section II briefly describes the related work of our study. Section III introduces the overall framework and technical details of our approach. Section IV describes the experimental settings and presents the experiment results. Section V concludes the paper and outlines future work.

II. RELATED WORK

A. Detecting the semantic relatedness between questions on programming community Q&A sites

Predicting the semantic relatedness between questions on programming community Q&A sites is beneficial to improve the efficiency of users in finding questions, and even helps them solve problems directly. Xu et al. [1] refer to a question and its answers in Stack Overflow as a knowledge unit, and divide these knowledge units into four classes: duplicate, direct, indirect, and isolated. In their paper, they proposed a CNN model, which adopts a convolutional neural network to extract contextual information in the knowledge units. They then calculate the semantic similarity of two contextual vectors by the cosine function and predict the class of knowledge unit pairs based on the semantic similarity. Fu et al. [8] proposed an SVM-based model (Tuning SVM), which uses word2vec [7] to obtain word embedding and adopts differential evolution (DE) as its tuning algorithm. In their study, tuning SVM with parameter tuning runs much faster than the CNN model. Following the study of Fu et al., Xu et al. [9] proposed the Soft-Cos SVM model in their paper. The Soft-Cos SVM model calculates the soft cosine similarity based on Simbow to measure the distance between knowledge unit pairs and adopt SVM as the final classifier. In addition, Xu et al. built a dataset containing 40,000 pairs of knowledge units. Based on the work of Xu et al., Shirani et al. [6] constructed a dataset with more than 300,000 pairs of knowledge units (hereafter, Knowledge Unit dataset). They also constructed two baseline models SOFTSVM and DOTBILSTM. SOFTSVM is similar to the previous SVM-based method, and it uses word2vec for word embedding of knowledge units, then calculates the cosine similarity, and finally predicts the semantic relatedness between knowledge units by SVM. DOTBILSTM uses BiLSTM to extract contextual information in the knowledge units, then calculates the inner product of the contextual vectors and obtains the probability distribution of each class using fully connected layers and softmax functions.

In recent work, Pei et al. [3] proposed the attention-based model ASIM. This model also adopts BiLSTM to encode local semantic information and obtain the interaction information between two knowledge units by soft attention mechanism. The experimental results show that ASIM is the state-of-the-art model in the dataset built by Shirani et al.. However, their approach was based on word-level interaction features between pairs of knowledge units through the soft attention mechanism. Therefore, these methods cannot sufficiently consider the semantic and interaction features at the sentence level, which is crucial for predicting the semantic relevance between questions. Therefore, we introduce a novel interaction method to improve the extraction of interaction features, hoping to achieve better performance in the task of predicting semantic relatedness between questions.

B. Duplicate question detection

Previous studies have shown [11] that the rapid growth in the number of duplicate questions is not conducive to website maintenance and it can lead to a decrease in the number of active users of community Q&A sites. AskUbuntu is another popular programming community Q&A site. Bogdanova et al. [14] construct a dataset with 30,000 question pairs from the AskUbuntu data dump and adopt CNN to detect duplicate questions. Based on the work of Bogdanova et al., Rodrigues et al. [15] released the AskUbuntu dataset’s clean version. They proposed a hybrid deep convolutional network model DCNN. The deep learning approaches described above do not require manually designed features and have a higher accuracy rate compared to traditional automatic detection methods. However, these methods also fail to consider the interaction information of sentences. We will use the AskUbuntu dataset released by Rodrigues et al. [15] to explore the generalization performance of DMSO in similar tasks.

III. THE APPROACH

According to Xu et al. [1], a question in Stack Overflow and its answers is a knowledge unit (KU). Based on the degree of relatedness between two knowledge units from high to low, the relatedness types between them can be defined as the following four classes:

- **Duplicate**: The questions in the two knowledge units are duplicate questions.
- **Direct**: Information in one knowledge unit can directly solve the question in another knowledge unit.
- **Indirect**: The information in one knowledge unit is helpful to the solution of the question in another knowledge unit.
unit, but the information alone cannot directly solve the question.
- **Isolated**: There is no semantic relatedness between the two knowledge units.

We treat this task as a multi-class classification problem. The input to the model is a pair of knowledge units, and the relatedness is predicted as one of the four classes mentioned above.

Figure 2 gives an illustration of the DMSO framework, which is mainly composed of the following five modules: (1) **Word Embedding Layer**, (2) **Word Encoding Layer**, (3) **Local Interaction Layer**, (4) **Global Interaction Fusion Layer**, and (5) **Prediction Layer**.

![Fig. 2 The framework of DMSO](Image)

**A. Input of the model**

We concatenate the title, body, and answers to the question to form a textual sequence as the input text. And we apply some data pre-processing steps on the input text, including dividing by sentence, normalizing URLs and numbers, removing punctuation marks and stop words, splitting camel case words, stemming, and changing all words to lowercase. We use the pre-processed text sequences as the final input to the model. Suppose the text sequence of a knowledge unit is \( KU = \{w_1, w_2, ..., w_n\} \), where \( n \) is the sequence length. The inputs of the model DMSO are text sequences \( KU_X \) and \( KU_Y \), and the target of the model predicts the relatedness of \( KU_X \) and \( KU_Y \). Since the two text sequences are treated symmetrically before the Prediction Layer, we will only present the processing of \( KU_X \) in the model to avoid repetition.

**B. Word Embedding Layer**

In this layer, we adopt word embedding techniques to convert words into their corresponding vector representations. The word embeddings pre-trained in the universal domain containing large amounts of data not related to software engineering, may lead to ambiguous representations of words [16]. Therefore, we use the title, body, and answers in Stack Overflow as the corpus, constructing a 300-dimensional word vector through word2vec [7]. Each word \( w_i \) in the knowledge unit is transformed into a vector representation \( x_i \) with 300 dimensions. Therefore, the knowledge units containing \( n \) words can be converted into corresponding matrix representation, which is the input of the word encoder of the model:

\[
KU_X = x_1 \oplus x_2 \oplus ... \oplus x_n \tag{1}
\]

Where \( \oplus \) is the concatenation operator.

**C. Word Encoding Layer**

In this layer, we adopt BiLSTM to fuse local features into each word’s original representation. BiLSTM is composed of a forward and a backward LSTM[4]. The semantic understanding of natural language words depends on the context. This means that the meaning of the word depends not only on what has been read before but also on what will be read. We adopt BiLSTM as the word encoder, which can capture local contextual dependencies from both forward and backward directions. Let \( \overrightarrow{h}_i, \overleftarrow{h}_i \in [1, n] \) denote the hidden state of the forward LSTM in the time step \( i \), and the encoding process of BiLSTM is as follows:

\[
\overrightarrow{h}_i = LSTM(\overrightarrow{h}_{i-1}, x_i), \forall i \in \{1, 2, ..., n\} \tag{2}
\]

\[
\overleftarrow{h}_i = LSTM(\overleftarrow{h}_{i+1}, x_i), \forall i \in \{1, 2, ..., n\} \tag{3}
\]

\[
X_i = [\overrightarrow{h}_i; \overleftarrow{h}_i], \forall i \in \{1, 2, ..., n\} \tag{4}
\]

**D. Local Interaction Layer**

Most of the previous deep learning models used in the semantic matching task on PCQA sites were based on representation models. These approaches do not consider the interaction information between these two questions. In the field of neural language processing, the attention mechanism was first applied in a neural machine translation model [19]. By applying the attention mechanism to the questions relatedness prediction task, the model can effectively extract the interaction information between questions through inter-sentence alignment and obtain better performance. Furthermore, Tay et al. [2] have shown that by equipping the attention mechanism with a more flexible structure, the model can generate more robust representations.

Therefore, in this Layer, we introduce a novel interaction method to improve the extraction of interaction features through constructing interaction attention and difference attention. Figure 3 shows the workflow of how the Interaction Feature Extractor extracts the interaction information between two features.

We first conduct a dot product between \( X_i \) and \( Y_j \), which are the output of two words in the knowledge unit \( KU_X \) and \( KU_Y \) from the Word Encoding Layer. After the weighted summation of the attention scores of all words in \( KU_X \) and \( KU_Y \), we can obtain the interaction attention matrix \( A \). And the difference attention \( D \) adopts a subtraction-based attention mechanism,
where $\cdot$ denotes the inner production operation, $\|X - Y\| \in \mathbb{R}^{n_{x} \times n_{y}}$, and $n_{x}$ is the $K |X|$ sequence length. We mask the two attention matrices and perform a $\text{sigmoid}$ operation on the difference attention $D$. Then we multiply $A$ and $D$ to get the attention matrix of the local interaction feature.

$$S^{X} = \text{softmax}(A \odot \text{sigmoid}(D)) \cdot Y$$  
(7)

$$S^{Y} = \text{softmax}(A \odot \text{sigmoid}(D)) \cdot X$$  
(8)

Where $\text{softmax}$ is a normalization function, $\text{sigmoid}$ is an activation function, and $\odot$ denotes element-wise multiplication.

### E. Global Interaction Fusion Layer

Previous methods only consider local interaction features and ignore the extraction of global features, so it is difficult to capture the semantic information and dependencies at the sentence level. To address the problem that local interaction features are insensitive to the understanding of global information, we extract the semantic features and interaction features of sentences to represent the global interaction features in knowledge unit pairs through the Global Interaction Fusion Layer. We refer to the method of converting word vectors to sentence vectors in the paper [5], which proceeds as follows.

$$G^{X} = \text{softmax}(W_{a}\tanh(W_{b}S^{X} + b))S^{X}$$  
(9)

Then we utilize the Interaction Feature Extractor to extract global interaction features, which can help capture the global relevance information between knowledge units. The global information complements the semantic relatedness that cannot be captured by local interaction.

$$F^{X}, F^{Y} = IFE(G^{X}, G^{Y})$$  
(10)

Where $IFE$ denotes the operation of the Interaction Feature Extractor, which is the same as Equations (5-8) in the Local Interaction Layer. We convert the aggregation features $F_{X}$ and $F_{Y}$ into fixed-size vectors.

$$V^{X}_{\text{pool}} = \text{Maxpooling}(F^{X})$$  
(11)

$$V^{Y}_{\text{pool}} = \text{Maxpooling}(F^{Y})$$  
(12)

Then we concatenate the vectors in the fusion layer.

$$T = \text{FFN}[V^{X}_{\text{pool}}; V^{Y}_{\text{pool}}; V^{X}_{\text{pool}} - V^{Y}_{\text{pool}}; V^{X}_{\text{pool}} \odot V^{Y}_{\text{pool}}]$$  
(13)

Where $\text{FFN}$ denotes a single-layer feedforward network.

### F. Prediction Layer

In the Prediction Layer, for the vectors $T$ obtained from the Global Interaction Fusion Layer, we adopt a single-layer feedforward network to get the feature vector. Finally, we calculate the probability distribution of four classes with the $\text{softmax}$ function.

### IV. EXPERIMENT

In this section, we conducted some experiments to answer the following research questions.

#### A. Research Questions

**RQ1: How effective is DMSO in predicting knowledge unit pairs of different classes?**

DMSO adopts the semantic interaction-based approach, introducing sentence-level interaction features to complement the semantic relatedness that cannot be captured by local interaction, which is a significant departure from previous work. To investigate the effectiveness of our approach, we compare the performance of DMSO with the baseline from Pei et al.’s work [3]. To present results more accurately, we keep the results of all models to three decimal places.

**RQ2: How much influence do the modules we proposed contribute to the improvement of DMSO?**

Two important modules we proposed, including Global Interaction Fusion Layer and Interaction Feature Extractor, can help DMSO to capture the rich features of knowledge units by extracting both the semantic and interaction features of sentences. To evaluate their contributions, we perform the following ablation studies, consisting in (1) removing the Global Interaction Fusion Layer (GIF) and replacing it with the max pooling operation; (2) removing the Interaction Attention (IA) in two Interaction Feature Extractors; (3) removing the Difference Attention (DA) in two Interaction Feature Extractors; and (4) removing two Interaction Feature Extractors (IFE). Then, we compare the revised model with the original model on the F1-score.

**RQ3: Does DMSO work well in other semantic relevance tasks from software engineering domains?**

Duplicate question detection in programming community Q&A sites is also a task to study semantic relevance in software engineering. In contrast to predicting question relevance, this task is a two-class classification problem where the evaluation metric is accuracy. We want to explore the generalization performance of DMSO through a different software engineering task.

---

Fig. 3  The architecture of the Interaction Feature Extractor

which allows the model to pay attention to dissimilar parts between knowledge unit pairs by element-wise subtraction. The difference attention is introduced to better capture fine-grained interaction information between knowledge units.

$$A = X^{T} \cdot Y$$  
(5)

$$D = \|X - Y\|$$  
(6)

Where $\cdot$ denotes the inner production operation, $\|X - Y\| \in \mathbb{R}^{n_{x} \times n_{y}}$, and $n_{x}$ is the $K |X|$ sequence length. We mask the two attention matrices and perform a $\text{sigmoid}$ operation on the difference attention $D$. Then we multiply $A$ and $D$ to get the attention matrix of the local interaction feature.

$$S^{X} = \text{softmax}(A \odot \text{sigmoid}(D)) \cdot Y$$  
(7)

$$S^{Y} = \text{softmax}(A \odot \text{sigmoid}(D)) \cdot X$$  
(8)

Where $\text{softmax}$ is a normalization function, $\text{sigmoid}$ is an activation function, and $\odot$ denotes element-wise multiplication.
B. Dataset

We carry out experiments on the dataset built by Shirani et al. [6]. This dataset focuses on Java-related knowledge units on Stack Overflow, and it contains 160,161 distinct knowledge units and 347,372 pairs of knowledge units with four types of relationships. In this paper, it is divided into three parts with the same proportion as in [6], that is, the training set containing 208, 424 knowledge unit pairs, the validation set containing 34, 736 knowledge unit pairs, and the test set containing 104, 212 knowledge unit pairs.

Moreover, to evaluate the generalization ability of DMSO in a similar task, we also run experiments on the duplicate question detection task in software engineering, using the clean version of the AskUbuntu dataset prepared by Rodrigues et al. [15]. In the AskUbuntu dataset, 24K question pairs are used for training, 6K for testing, and 1K for validation. The two classes in the AskUbuntu dataset are balanced, so there are an equal number of duplicate and non-duplicate question pairs. Since the dataset contains only questions and not answers, we concatenate the question titles and bodies as inputs to the model and modify the output of the prediction layer to 2 classes. The other parts of the model remain unchanged.

C. Evaluation Metric

We use the same evaluation metrics as in the previous work [3, 6] to evaluate DMSO's performance. Precision$_i$ represents the proportion of knowledge unit pairs correctly classified as class $i$. Recall$_i$ is the percentage of all class $i$ correctly classified. F1-score$_i$ is harmonic mean of Precision$_i$ and Recall$_i$.

\[
\text{Precision}_i = \frac{\text{TruePositive}_i}{\text{TruePositive}_i + \text{FalsePositive}_i} \quad (14)
\]
\[
\text{Recall}_i = \frac{\text{TruePositive}_i}{\text{TruePositive}_i + \text{FalseNegative}_i} \quad (15)
\]
\[
F1 - \text{score}_i = \frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \quad (16)
\]

D. Implementation Details

The deep learning framework we use is PyTorch 1.9.1 with CUDA 10.2 as the GPU computing platform, and all experiments are implemented in a python 3.7 environment. For our model, 300-dimensional word embeddings are used according to experience. We set the hidden size as 200 for all feedforward layers, and select GeLU [17] as the activation function of the feedforward network. Adam [10] optimizer with an initial learning rate of 0.001 was applied. The dropout strategy [18] is adopted to avoid overfitting and the dropout rate is set to 0.2. We use a batch size of 32. The model is trained for 30 epochs to minimize the cross-entropy loss. We release the source code of DMSO and hope to facilitate future research.

https://github.com/anonymousseke/dmso

E. Result

RQ1: How effective is DMSO in predicting knowledge unit pairs of different classes?

Table I presents the experiment results. The best results are highlighted in bold. We can see that the results for the direct and indirect classes are much lower than the duplicate and isolated classes, indicating that the semantic information in these two classes is more confusing for understanding. So that SOFTSVM and DOTBLSTM are difficult to identify patterns for classification, which leads to poor results (34.7%, 50.6%, 55.2%, and 61.0%). This result reflects that this class limits the overall performance of all approaches included DMSO.

Meanwhile, compared to feature-based SOFTSVM, deep learning methods do not rely on fixed patterns and allow advanced abstraction modeling of text to understand the relevance of two knowledge unit pairs and obtain better scores. We can see that DMSO achieves the best results for all four classes, with 83.4%, 83.2%, and 83.3% in overall Precision, Recall, and F1-score. In addition, compared to the state-of-the-art model ASIM, DMSO improves by 1.5% in F1-score, and by 1.3% and 1.5% in both Precision and Recall. This result shows that extracting semantic and interaction features of sentences is important for understanding and identifying the relevance of knowledge unit pairs, especially in long text sequences and complex expressions in software domains.

RQ2: How much influence do the modules we proposed contribute to the improvement of DMSO?

The experimental results are shown in Table II. After removing GIF or DA, respectively, the revised models’ performance decreases to some degree (1.7% and 1.0%, in terms of F1-score, respectively). However, after removing the IA or IFE, the model’s performance decreases more obviously (2.7% and 4.5%, in terms of F1-score, respectively), especially in predicting the direct class (10.3% and 2.5% in terms of F1-score, respectively). This experiment’s results prove the importance of the Interaction Feature Extractor and the Global Interaction Fusion Layer, which plays an essential role in predicting question relatedness.

RQ3: Does DMSO work well in other semantic relevance tasks from software engineering domains?

Table III presents the results. DMSO outperforms other models, including rule-based approaches (Jcrd [12]), classifiers (SVM-bas [14], SVM-adv [15], and SOFTSVM), and neural networks (CNN [14], DNN [13], DCNN [15], DOTBLSTM, and ASIM), achieve an accuracy of 97.69%. Moreover, DCNN is the best model in the paper [15] on the AskUbuntu dataset, and the ASIM is the state-of-the-art model on the Knowledge Unit dataset. The experimental results show that DMSO has a degree of generalization capability that can be applied to other software engineering fields.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a deep learning model DMSO we introduce a deep learning model (DMSO) to predict semantic relatedness between questions in Stack Overflow, helping developers acquire related knowledge to solve the programming
issues. The experiment results show DMSO’s effectiveness and consistency in predicting question relatedness, outperforming some baseline models in previous works. Moreover, in the duplicate question detection task of AskUbuntu, DMSO can also achieve state-of-the-art performance, which proves its generalization ability. We will explore the implementation of an application that provides related questions when users retrieve questions or post new issues in future work.

REFERENCES


### Table I

Comparing the result (Precision, Recall, and F1-score) of DMSO and the baseline models in each relatedness class.

<table>
<thead>
<tr>
<th>Model/Classes</th>
<th>Duplicate</th>
<th>Direct</th>
<th>Indirect</th>
<th>Isolated</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSO-GIF</td>
<td>0.934</td>
<td>0.739</td>
<td>0.704</td>
<td>0.941</td>
<td>0.833</td>
</tr>
<tr>
<td>DMSO-IA</td>
<td>0.916</td>
<td>0.714</td>
<td>0.689</td>
<td>0.932</td>
<td>0.823</td>
</tr>
<tr>
<td>DMSO-DA</td>
<td>0.915</td>
<td>0.716</td>
<td>0.685</td>
<td>0.932</td>
<td>0.819</td>
</tr>
<tr>
<td>DMSO-IFE</td>
<td>0.912</td>
<td>0.636</td>
<td>0.678</td>
<td>0.925</td>
<td>0.788</td>
</tr>
</tbody>
</table>

### Table II

Comparing the F1-score of DMSO and the revised models.

<table>
<thead>
<tr>
<th>Model/Classes</th>
<th>Duplicate</th>
<th>Direct</th>
<th>Indirect</th>
<th>Isolated</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSO-GIF</td>
<td>0.949</td>
<td>0.739</td>
<td>0.704</td>
<td>0.941</td>
<td>0.833</td>
</tr>
<tr>
<td>DMSO-IA</td>
<td>0.933</td>
<td>0.715</td>
<td>0.679</td>
<td>0.935</td>
<td>0.816</td>
</tr>
<tr>
<td>DMSO-DA</td>
<td>0.934</td>
<td>0.733</td>
<td>0.689</td>
<td>0.932</td>
<td>0.823</td>
</tr>
<tr>
<td>DMSO-IFE</td>
<td>0.912</td>
<td>0.636</td>
<td>0.678</td>
<td>0.925</td>
<td>0.788</td>
</tr>
</tbody>
</table>

### Table III

Performance of different models in the Askubuntu dataset.

<table>
<thead>
<tr>
<th>Model/Classes</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>0.790</td>
</tr>
<tr>
<td>CNN</td>
<td>0.785</td>
</tr>
<tr>
<td>SVM-adv</td>
<td>0.7587</td>
</tr>
<tr>
<td>SVM-bas</td>
<td>0.7025</td>
</tr>
<tr>
<td>ASIM</td>
<td>0.9625</td>
</tr>
<tr>
<td>DMSO</td>
<td>0.9769</td>
</tr>
</tbody>
</table>
Learning Event Logic Graph Knowledge for Credit Risk Forecast

Yifan Wang * Jibin Wang * Jing Shang * Zhuo Chen * Xuelian Ding*
Heyuan Wang†
* China Mobile Information Technology Co., Ltd.
† School of Computer Science, Peking University

Abstract

With the development of event knowledge graph technology, researchers have solved the singleness problem of event graph based on temporal relationship by constructing event logic graph, but have not integrated the multiple relationships among events with time series data for trend prediction. In addition, due to the impact of COVID-19, corporate credit risks have been gradually exposed in recent years, and defaults have occurred frequently. The technology of event graph and event logic graph is mostly used for event schema induction, script induction, etc., but abundant graph knowledge is not well exploited for forecast task.

To fill this gap, we construct an event logic graph by extracting various types of event relationships, such as causal relationship, sequential relationship, parallel relationship, and reversal relationship. Different types of edges among events are used to represent different relationships. Combined with the time series of corporate credit bonds, a temporal convolutional network driven by event logic graph is built, and applied to forecast corporate credit risk.

We extract structured events from financial news, construct event logic graph and learn the graph knowledge. Then, the event logic graph embedding is combined with time series of bonds to forecast whether the corporate will default. Experiments show that the proposed method outperforms baseline methods in forecasting credit risk.

1 Introduction

Events contain a large number of internal composition structures (such as participants, time, place, etc.) and external associations (such as causal relations, sequential relations, parallel relations, reversal relations, etc.). The news information platform has accelerated the spread of information among various social groups. How to perceive hot events and sort out the context among them has become the key to tracking sudden social turmoil, epidemic diseases, credit defaults and other events.

Most of the existing intelligent news system construct the event graph based on temporal relationship, but there are complex internal and external correlations among events. The cascade reactions brought about by different event relationships are disparate. We can assist decision-making in many fields such as natural disasters, aviation safety, and public health, in the method of extracting events and identifying relationships from news texts.

There are three key technologies in applying event graph to forecasting: event detection, event relationship extraction, and trend forecast.

In terms of event detection, topic detection and text clustering are traditional technical methods, using BoW model and LDA model to represent the topic of the text. These methods have the following problems: 1. Clustering algorithms need to determine the number of clusters in advance to achieve event classification. 2. The unbalanced distribution of event clusters increases the difficulty of clustering.

In terms of event relationship extraction, the traditional solution is to implement topic discovery through article clustering, and construct the relationship among events based on sequence of events. However, how many relationship types among events is still a controversial issue, and identifying different relations from text is challenging.

In terms of trend forecast, most of the current research based on event context is applied to smart news in the aim of assist the public in visualizing hot spots. A small number of financial quantification teams extract structured event tuples from financial news, and use knowledge graph technology to correlate discrete event tuples with each other. Event embeddings are obtained by training event tuples and knowledge graph triples. On this basis, the method of multichannel connection is adopted, the price vector and event embedding are used as the input of model to forecast the stock price. However, such methods do not use the multiple relationship among events for forecast.

Our contribution can be summarized as follows:

Extract multiple event relationships, such as sequential relationships, causal relationships, reverse relationships and parallel relationships. Construct event logic graph using events as nodes and multiple event relationships as edges.

Build a temporal convolutional network model driven by event logic graph, which integrates news events and corporate credit bond transaction data to forecast credit risks. The comparative experiments and ablation experiments show that our approach outperforms the baseline methods, and multiple relationship has better performance in trend forecast than the single relationship.

DOI reference number: 10.18293/SEKE2023-222


text: ntegrated in entity-centric knowledge graphs and artificially curated semi-structured resources. The proposal of EKG promotes a global view of events and temporal relationships. Ding et al.[13] proposed Event Logic Graph(ELG). Nodes in ELG represent events, and edges represent relationships among events. ELG reveals the development and evolution process of objective events.

Zhang et al.[14] proposed a large-scale event knowledge graph (ASER) for discovering real-world activity knowledge. ASER defines a brand new knowledge graph, in which each vertex is the basic element of an event, and the event relationship is the connection a hyperedge of several vertices. In addition, some researchers try to construct commonsense knowledge graphs around events, such as Event2Mind, GLUCOSE, ATOMIC.

2.4 Applications in Trend Forecasting

The event graph is centered on events, describing event information and the relationship among events. Based on this, technologies such as event prediction and trend prediction are realized. By analyzing the development process of historical events, it is possible to predict future events. Many researchers use contextualized events such as event skeletons in different fields to predict trends, such as the evolution process of natural disasters, event evolution analysis, stock price prediction, etc.

3 Method

3.1 Explicit Event Relationship Extraction

We unify the extraction methods of various event relationships to make the processing of news corpus more efficient. Inspired by Chang et al., this paper adopts the combination of explicit relationship trigger words and pattern recognition. Different from Chang’s approach, in addition to establishing temporal relations, this paper formulates explicit trigger words and linguistic rules for causal, parallel, and reversal relations. We set different types of relational triples (preword, postword, type) for pattern matching and event tuple constructing, where preword and postword do not necessarily appear at the same time, such as (“due to”, “resulting in”, causal).

We constructed the corresponding syntactic patterns, and extracted event tuples (text, type, preword, prepart, postword, postpart) of various relations. "type" is the type of event relationship, "text" is the original text, "preword" and "postword" represent the event relationship trigger words obtained by pattern matching, the prepart and postpart are the event content before and after the relationship trigger words.

2 Related Work

2.1 Event Extraction

The research and application of event extraction mainly rely on machine learning and deep learning. Based on the existing models, the following two common problems are summarized:

1. How to learn the semantic representation of events from the given text: mining effective features is the key to the model. Early methods designed fine-grained features such as lexical, syntactic, and kernel-based features. Neural networks have been tried in this task, including CNNs and Transformers. Due to the complexity of event structures, recent studies have begun to use additional information such as entities, document-level information, and syntactic structures.

2. How to extract events across sentences or at document level: Current research mainly focuses on the sentence level, with the basic assumption that events are manifested in sentences. Compared with sentence-level event extraction, document-level event extraction needs to consider more complex issues: including parameter dispersion, multi-event expression, etc.

2.2 Event Relation Extraction

Event Co-reference Resolution is to confirm whether multiple event elements belong to the same event, which is treated as a classification or ranking problem. Machine learning models are widely used in this field, such as decision tree classifier[1], information propagation models[2] and multi-loss neural models[3]. These models focus on understanding the context of two events.

Event Causality Extraction is often viewed as a classification task. Existing models generally complete classification under the premise of supervision, the key point is how to extract clues and how to represent the semantics of causality. Extracting effective clues of contextual event causality requires the use of various text features, including syntactic features[4], lexical features, explicit causal patterns[5], statistical causality, etc.

Temporal Relation Extraction is mainly based on the TimeML format[6], and most methods solve it as a classification problem. Early ETE models usually relied on temporal rules[7]. Some researchers also used temporal context features to build models and extract temporal relationships based on machine learning[8]. In addition, neural network models are widely used, such as the classic CNNs, LSTMs methods[9] and the BiLSTM model based on dependent paths[10]. In addition, some more refined improved methods achieve more accurate extraction results[11].

2.3 Event Knowledge Graph

Gottschalk and Demidova[12] designed and implemented the Event Knowledge Graph(EKG). At present, events and their temporal relationships are mostly distributed in entity-centric knowledge graphs and artificially curated semi-structured resources.
3.2 Construction of Event graph

The event graph we constructed uses events as nodes and the relationship among events as edges. Both “event” and “relation” are obtained from the event-relation tuple.

Take the news of ICBC as an example, “Regulations have tightened the reporting standards, leading to a general decline in the stock of public offerings sold by securities companies.” The event relationship tuple can be obtained through the explicit event relationship extraction method (type=“causal”, preword=“null”, prepart=“Regulations have tightened the reporting standards”, postword=“leading to”, postpart=“The stock of public offerings sold by securities companies has generally dropped”). Then set an edge whose type is “causal” between the two events. The construction methods of sequential relationship, parallel relationship and reverse relationship are the same.

3.3 ELGTCN

When forecasting the credit risk of corporates, it is also necessary to combine with financial time series data, such as the lowest transaction price of bonds. This paper uses Event Logic Graph Driven Temporal Convolutional Networks (ELGTCN) to combine event logic graph with bond data in corporate credit risk forecast.

3.3.1 Representation of Original News

For the original news, we use word2vec to encode and obtain vector sequences. Calculate the maximum value of each dimension through the maximum pooling operation to capture the most significant attributes in the sequence. The average value of each vector sequence is calculated by the average pooling operation to obtain universal information. The max pooling and average pooling operations on vector sequences are two complementary ways. Finally, the results of the two operations are connected in parallel to obtain the representation of mean-max. The calculation process is as follows:

![Figure 1. ERCDTN](image1)

\[ z_{\text{max}}[i] = \max_t h_t^e \]
\[ z_{\text{mean}} = \frac{1}{T} \sum_t h_t^e \]
\[ z = [z_{\text{max}}, z_{\text{mean}}] \]
\[ z_{\text{max}}^d = z_{\text{max}} \odot \sigma (W : z_{\text{max}} + b) \]
\[ z_{\text{mean}}^d = z_{\text{mean}} \odot \sigma (W : z_{\text{mean}} + b) \]
\[ h_t^d = z_{\text{max}}^d \oplus z_{\text{mean}}^d \]

3.3.2 Representation Method of ELG

We transform the multiple relationship graph structure into a low-dimensional space vector by means of TransE[15]. Suppose there is an event relation triplet of \((e_1, \text{relation}, e_2)\), where the vector of event \(e_1\) is expressed as \(V_{e_1}\), event \(e_2\) The vector representation of \(V_{e_2}\), the relationship between event pairs is \(V_{\text{relation}}\). The ultimate goal is to map events and their relationships into \(k\) dimensional vectors. The essential idea is that if there is a relationship \(\text{relation}\) between the event pair \(e_1\) and \(e_2\), then try to make \(V_{e_1} + V_{\text{relation}} \approx V_{e_2}\), otherwise, keep as far away as possible.

In order to realize the representation learning of multiple relations, the relation \(r\) is modeled as a hyperplane \(w_r\), and a transfer vector \(d_r\). First, project the two events on the square of \(w_r\), and then calculate the distance on the projected hyperplane. That is, \(V_{r_1}(e_1) = e_1 - w_r^t e_1 w_r\), \(r\) is expressed as a transition vector \(d_r\), \(V_{r_2}(e_2) = e_2 - w_r^t e_2 w_r\). Use the Euclidean distance between \(V_{r_1}(e_1) + d_r\) and \(V_{r_2}(e_2)\) as the distance between two events.

Multivariate event relations are projected to different hyperplanes, and different relations of events are represented by vector calculations between different hyperplanes. Set the distance function \(d(V_{r_1}(e_1) + V_{r_1}(e_2))\), for all event pairs in the event logic graph, minimize the distance function \(d(V_{r_1}(e_1) + V_{r_1}(e_2))\). Specifically, the loss function \(\mathcal{L}\) is defined as follows, where \(r_i\) represents different event types, \(i \leq 4\).

\[ \mathcal{L} = \sum_{(e_1, r, e_2) \in S} \sum_{(e_1', r, e_2') \in S'} \left[ \lambda + d(V_{r_1}(e_1) + V_{r_1}(e_2)) - d(V_{r_1}(e_1') + V_{r_1}(e_2')) \right]_+ \]
Where \( [x]_+ \) represents the positive part of \( [x] \), \( S'_{(e_1,r,e_2)} \) represents the relational tuple whose head or tail is randomly replaced. Increase the distance function of non-existent event-pair relations by constructing error samples. The composition of \( S'_{(e_1,r,e_2)} \) is as follows:

\[
S'_{(e_1,r,e_2)} = \{(e'_1, relation, e_2) \mid e'_1 \notin E\} \\
\cup \{(e_1, relation, e'_2) \mid e'_2 \notin E\} \tag{3}
\]

### 3.3.3 Representation of Time Series Data Related to Corporate Credit Debt

According to financial market experience, bond variables, corporate variables and macro variables have a certain early warning effect on corporate credit risk. Bond variables include minimum transaction price, coupon rate, etc. Corporate variables include return on total assets, proportion of long-term liabilities, etc. Macro variables include risk-free interest rates and the CSI 300 Index. We use the above information as time series data reflecting corporate credit risk. By splicing the vector representations of the above three kinds of data, the original news \( \mathcal{X} \), the event logic graph \( \mathcal{G} \), and the time series data \( \mathcal{X}' \) are fused (as shown on the right side of Figure 1), as the input for forecast.

### 3.3.4 ELGTCN for Forecast

Inspired by Bai et al. [16], the TCN model is used to prevent the look-ahead error that may exist in the application of time series data. Longer historical data can be looked back through deep networks augmented with dilated convolutions and residual layers.

We employ a 1D fully convolutional network (FCN) architecture, where each hidden layer has the same length as the input layer, and zero padding is added to keep subsequent layers the same length as the previous layer. In this way, the network can produce an output of the same length as the input. In addition, TCN uses causal convolution, and the output at time \( t \) is only convolved with elements at time \( t \) and earlier in the previous layer, so that there is no look-ahead bias.

The historical information that causal convolution can recall is linear with the depth of the network. The prediction of enterprise credit risk is a task that requires a long historical review. Therefore, we use the expansion convolution to achieve an exponentially large receptive field. Specifically, for a one-dimensional sequence input \( x \in \mathbb{R}^n \) and a filter \( f : \{0, \ldots, k-1\} \rightarrow \mathbb{R} \), the dilated convolution operation \( \mathcal{F} \) on element \( s \) in the sequence is defined as:

\[
\mathcal{F}(s) = (x * df)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s - d \cdot i} \tag{4}
\]

where \( d \) is the dilation factor, \( k \) is the filter size, and \( s - d \cdot i \) represents the direction of the history. Therefore, the dilation factor is equivalent to introducing a fixed step size between every two adjacent filters. If the size of \( d \) is 1, the expansion convolution is an ordinary convolution kernel; if a larger expansion factor is used, the output result can represent a larger range of input data, that is, the receptive field of the convolutional network is larger.

Deep networks are prone to the problem of gradient disappearance or gradient explosion. At present, BN, regularization and other methods can be used to improve it, but it still cannot support too deep networks. Therefore, we use the residual module to realize the identity mapping of cross-layer connections, and learn the residual function \( F(X) = H(X) - X \), that is, to learn the partial modification of the input \( X \). Introducing the residual module can solve the problem of gradient disappearance.

The input of the ELGTCN model includes: bond-related time series data \( \mathcal{X} \) (bond variables, corporate variables, macro variables), original news \( \mathcal{N} \) and event logic graph \( \mathcal{G} \). The data is normalized and mapped to a vector representation, where each vector \( p_t \) represents the fusion vector of the time series data, news corpus, and event logic graph of the bond trading day \( t \).

\[
\mathcal{P} = \{p_0, p_1, \cdots, p_{t-1}\} \tag{5}
\]

Credit risk prediction can be abstracted as a binary classification problem. Considering that the liquidity of bonds is weaker than that of stocks, we choose to use one week as the step size. We use historical \( n \) days of company news and financial data to predict whether the company in the specified target \( \mathcal{S} = \{s_1, \ldots, s_N\} \) has credit risk.

### 4 Experiment

#### 4.1 Datasets and Compared Methods

The data required for the experiment include financial data and news texts, and the access methods include: wind database and news websites. All industrial credit bonds in China from January 2014 to December 2021 were extracted from the wind database, with a total of 36,702 samples. The financial news that can be periodically obtained through the Scrapy crawler framework involves more than 4,000 listed companies. Bond variables include the minimum transaction price, coupon rate, etc.

<table>
<thead>
<tr>
<th>Model</th>
<th>Raw Data</th>
<th>Training Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCN</td>
<td>( \mathcal{X} )</td>
<td>TS vector + event embedding (^1)</td>
</tr>
<tr>
<td>PVEB-TCN</td>
<td>( \mathcal{X} + \mathcal{N} )</td>
<td>TS vector + event embedding (^2)</td>
</tr>
<tr>
<td>TDPVEB-TCN</td>
<td>( \mathcal{X} + \mathcal{N} + \mathcal{E} )</td>
<td>TS vector + relation embedding</td>
</tr>
<tr>
<td>KDTCN</td>
<td>( \mathcal{X} + \mathcal{N} + k )</td>
<td>TS vector + knowledge embedding</td>
</tr>
<tr>
<td>ELGTCN</td>
<td>( \mathcal{X} + \mathcal{N} + \mathcal{G} )</td>
<td>TS vector + relation embedding</td>
</tr>
</tbody>
</table>

\(^1\) TS vector + event embedding
\(^2\) TS vector + relation embedding
callability, putability, etc. Company changes include return on total assets, proportion of long-term liabilities, company size, whether it is a listed company, etc. Macro variables include risk-free interest rate, CSI 300 index.

4.2 parameters and comparison settings

Set time series data as $X$, news text as $N$, event knowledge graph as $K$, event graph as $G$. We select several commonly used baseline models and set the data types they contain as shown in Table 1.

<table>
<thead>
<tr>
<th>Table 2. Evaluation Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>TCN</td>
</tr>
<tr>
<td>PVEB-TCN</td>
</tr>
<tr>
<td>TDPVEB-TCN</td>
</tr>
<tr>
<td>KDTDCN</td>
</tr>
<tr>
<td>ELGTCN</td>
</tr>
</tbody>
</table>

The baseline model has different inputs. In the first column, EB means event embedding, PV means bond time series data vector, TD means event graph driven, KDTDCN is a stock prediction method driven by knowledge graph, ELGTCN is proposed for this paper model. Event embedding 1 means that the event structure is extracted from the text and used directly for trend forecasting, event embedding 2 is used for trend forecasting after the time context of the event is generated, event knowledge embedding is the embedding of knowledge graph constructed by event elements, and event relation embedding is the embedding of event logic graph.

The training process uses the Adam optimizer. In order to avoid model overfitting, set the Batch Size to 5, the learning rate lr to $1 \times 10^{-3}$, and the Dropout Rate to 0.3.

4.3 Experimental Results

4.3.1 Visualization of Event Logic Graph

Taking "Shanghai Electric" as an example, we use hot news in 2021 to generate an event logic graph. It can be seen in Figure 3 that there are many examples of causal and reverse relationship, which is common in listed companies.

We visualized Liu et al. [17]'s event relationship extraction method based on knowledge enhancement. As shown in Figure 4, this method realizes the identification and extraction of causality, but the type of event relationship is relatively single and covers less content.

Comparing Figure 3 and Figure 4, it can be found that the method in this paper has advantages over the control method in terms of the diversity of event relationships, the comprehensiveness of event scope, and the integrity of event development.

4.3.2 Analysis of Evaluation Metrics

We compare the performance of the models under different inputs, and calculate the accuracy, precision, recall and F1-score of each model.

It can be seen from Table 2, PVEB-TCN incorporates the event information in the news corpus, and the performance of the model has improved. KDTDCN builds a knowledge graph based on event elements and combines bond data for prediction, whose performance is lower than other models. ELGTCN has the best performance.
In order to reflect that multiple relationships are more advantageous than single relationships, we separately use sequential relationships, causal relationships, parallel relationships, and reversal relationships as the basis for trend prediction. Table 3 shows the best performance and worst performance of single event relationship and compound event relationship in forecast. Through the experimental results, it can be found that the indicators of ELGTCN that integrates multiple event relationships are the best.

5 Conclusion

We design temporal convolutional networks driven by event logic graphs (ELGTCN). We constructed a graph of event by identifying multiple relationships, and applied it to the forecasting of corporate credit risk. First, we used a combination of relational triggering and pattern matching to extract multiple event relations from news text. Secondly, when constructing the graph structure, events were regarded as nodes and event relations were regarded as edges. Finally, ELGTCN was designed to integrate the event logic graph with time series data, used for corporate credit risk forecast. Experiments show that the method outperforms baseline methods in forecasting default risks.

References


SAWD: Structural-Aware Webshell Detection System with Control Flow Graph

Junmin Zhu†§, Yizhao Yao†§, Xianwen Deng†§, Yaoguang Yong‡, Yanhao Wang††,
Libo Chen†, Zhi Xue†∗, and Ruijie Zhao†∗
†Shanghai Jiao Tong University, Shanghai, China
‡Tencent Technology (Shanghai) Company Limited, Shanghai, China
††QI-ANXIN, Beijing, China
§Equal Contribution
∗Corresponding Authors

Abstract

With the increasing prevalence of web servers, protecting them from cyber attacks has become a crucial task for online service providers. Webshells, which are backdoors to websites, are commonly used by hackers to gain unauthorized access to web servers. However, traditional methods for detecting webshells often fail to produce satisfactory results due to the use of obfuscation or encryption to conceal their characteristics. In recent years, webshell detection methods based on deep learning (DL) have received significant attention, but they struggle to preserve the syntax and semantic information contained in the source code. In this paper, we propose a structural-aware webshell detection system to address these problems, denoted as SAWD. Specifically, we first generate the control flow graph (CFG) with syntax and semantic information from the PHP source code. Then, we leverage CFG to build our graph representation, which consists of the adjacency matrix and keywords-based basic block features. Finally, based on our graph representation, we adopt convolutional neural networks (GCN) combined with graph pooling to detect webshells more efficiently. Experimental results demonstrate that our method outperforms state-of-the-art webshell detection systems on the collected dataset.

Index Terms—Webshell detection, deep learning, control flow graph, graph neural networks.

I. Introduction

With the development of the internet, web servers have become a prime target for hackers. Hackers often use webshell, a web script that contains a malicious code fragment, to launch web attacks for system intrusion. Once the web server is compromised, a vast amount of sensitive user information stored in it will be stolen, causing irreparable damage to the service provider. Thus, it is critical to design an effective webshell detection system to protect system security.

Currently, webshell detection systems are divided into three categories, namely traffic-based detection, log-based detection, and file-based detection. Traffic-based detection involves real-time monitoring of server communication traffic and determining if it is a request communicating with a webshell through pre-set rules [1]. Unfortunately, it is inefficient to detect webshell behavior from large-scale traffic. Log-based detection examines whether the HTTP requests and responses recorded on the website contain malicious behavior for webshell detection [2]. However, this system can only audit the behavior during or after an attack, which means the system has been hacked. Compared with the above two solutions, File-based detection is more efficient for finding potential threats before attackers launch the attack. It analyzes the code within program files to extract functional and logical characteristics for detecting webshells.

Traditional file-based webshell detection methods mainly rely on building feature libraries based on file attributes (e.g., file name, file creation time, file modification time, and high-risk functions), but they may not be applicable to webshell files that have been encrypted or obfuscated, which are beyond the scope of the established feature libraries. Researchers then suggest using statistical features like information entropy, longest word, and overlap index to identify obscured webshell files. However, they have limited applicability as well as low detection performance. Another mainstream webshell detection method is to analyze the data dependencies between variables through syntax and semantic analysis. Since it requires manual setting of pollution sources, pollution recipients,
and pollution propagation rules, it may become difficult to detect the webshell using new PHP features. Recently, webshell detection methods based on deep learning (DL) have received significant attention. They first convert the source code to the opcode sequence, and then adopt deep neural networks for feature extraction. However, this method often disregards the operands within the opcode, thereby overlooking critical information contained therein. So this approach struggles to fully capture the syntax and semantic information of PHP source code.

In this paper, to solve the above challenges, we propose a **Structural-Aware Webshell Detection system (SAWD)**, which generates graph representation for modeling PHP source code and leverages DL-based graph networks for feature extraction. Specifically, we first generate the control flow graph (CFG) that can preserve the syntax and semantic information of PHP source code. Then, we convert the CFG into the adjacency matrix and extract the feature of the basic block by counting the most frequent keywords, which plays a significant role in detecting obfuscation and encryption behavior. At last, we use graph convolutional neural networks (GCN) combined with MinCutPool as our structural-aware model for webshell detection. GCN is used to extract features from a topological graph, then we utilize the graph pooling method to cluster the nodes based on the similarity of the basic blocks that perform the same function in the CFG, thus improving the generalization ability. The major contributions of the proposed work are three-fold:

- To generate the effective graph representation with the syntax and semantic information, we first convert the PHP source code to CFG. Then, we adopt the keywords-based method to represent each basic block and the adjacency matrix to capture the relationship between basic blocks.
- We develop a structural-aware webshell detection model using GCN combined with graph pooling, which can leverage our graph representation for more efficient code analysis. To our best knowledge, this is the first investigation in this direction.
- We evaluate SAWD on our collected dataset with three webshell types i.e., big trojan, small trojan, and one-word trojan. Experimental results demonstrate that our method outperforms state-of-the-art webshell detection systems.

## II. Related Work

To efficiently detect encrypted webshell, researchers proposed to use file-based webshell detection systems. Some work identifies obfuscated and encrypted content in scripts (e.g., NeoPI [3]), but the design of statistic features in their methods relies heavily on expert knowl-
edge. In recent work, more and more studies adopt deep learning algorithms for webshell detection, which can be categorized based on the features used as source code-level methods, opcode-level methods, and AST-level methods. Li et al. [4] utilized Word2Vec for the vectorization of PHP source code and gated recurrent unit (GRU) for effective detection. Zhao et al. [5] converted PHP code into opcode sequences and incorporated TF-IDF-based weighted processing techniques, utilizing an XGBoost model for classification. Kang et al. [6] proposed an RF-GBDT model that employed an improved TFIDF-chi feature to extract webshell opcode features. Additionally, they combined statistical features and opcode sequences of PHP files to enhance detection efficiency. Although using opcodes as features can reduce the impact of techniques such as obfuscation on detection performance, they do not preserve syntax and semantic information present in the original code. Li et al. [7] presented Shellbreaker which uses static analysis and machine learning techniques to extract features from PHP scripts for detection. However, this approach is vulnerable to failure when identifying new webshells, especially when attackers use advanced encoding strategy (e.g., letter slicing). Cheng et al. [8] proposed MSDetector to parse PHP scripts into an AST and extract lexical tokens. Although using AST as a feature effectively preserves syntax and lexical information present in source code, semantic information is challenging to retain completely.

In the field of program analysis, graph-based methods have become increasingly common. Reps [9] first proposes to convert some program analysis problems into graph reachability problems, and solve them using graph reachability algorithms. Yamaguchi et al. [10] introduced the code property graph, which combines the properties of abstract syntax trees, control flow graphs, and program dependence graphs to create a novel representation of source code. Backes et al. [11] applied this approach to PHP and used taint analysis to discover vulnerabilities in PHP code. However, there is currently no work on detecting webshells based on graphs. Additionally, graph neural networks can utilize the information of nodes and edges in a graph to learn representations of nodes, greatly enhancing their ability to preserve the complete syntax and semantic information of source code. All these methods have inspired us to adopt graph-based methods and graph neural network models to address the problem of incomplete preservation of syntax and semantic information in previous work.

## III. Method

We describe the detailed designs in this section. The overview of our proposed method is illustrated in Fig. 1.
A. Graph Representation of PHP Source Code

1) Graph Construction: We employ PHP-CFG, an open-source project developed and maintained by Anthony Ferrara, to establish the CFG from the PHP source code. It first parses PHP code into an Abstract Syntax Tree (AST), which contains all the syntax information in a program. To further capture the semantic information, PHP-CFG then iterates through the AST nodes to group adjacent AST nodes in the same code branch into a basic block. Finally, all the blocks are connected by directed edges according to the execute sequence and conditional jumps, and the directed CFG is established, which precisely provides a clear representation of the syntax and semantic information. The variables and function calls in the basic code block represent the syntax information, while the connection relationships between blocks, i.e., the execution sequence and conditional jumps, represent the semantic information.

Fig. 2 illustrates an example of a webshell and its corresponding CFG, where the CFG consists of four nodes and three edges. Each node corresponds to a basic code block, and each edge represents a conditional or loop relationship in the PHP code. For the PHP source code, PHP-CFG first groups the conditional AST node and its preceding nodes in the same code branch into a basic block. In addition to normal basic blocks, a main code block is automatically created as the entry point during the initial phase. Starting from the main code block, PHP-CFG traverses all blocks and establishes “if-else” relationships as edges for connecting these code blocks.

We then convert the directed CFG into the digitized adjacency matrix, which enables us to quickly identify the entry and exit points of a CFG, as well as its loops and branches. Moreover, the symbolic adjacency matrix can be used to perform various graph-based algorithms. Specifically, we first generate an $N \times N$ zero matrix, where $N$ is the total number of blocks in a CFG. This matrix serves as a template for capturing the relationship between basic blocks. If there is a directed edge between any two basic blocks, for example from block $m$ to block $n$, we set the corresponding entry in the matrix to 1. To illustrate this, let us continue to consider the CFG in Fig. 2, which consists of five block nodes labeled n1, n2, n3, n4, and n5. Note that n2 has three adjacent blocks: n1, n3, and n4. The control flow goes from n1 to n2, then from n2 to either n3 or n4. To represent these connections, we set entries (1, 2), (2, 3), and (2, 4) in the adjacency matrix to 1, while (2, 1) remains 0. Similarly, we apply the same procedure to all other basic blocks in the CFG. The conversion result
is shown in Fig. 1 Step 1. By doing so, we obtain an adjacency matrix to visualize the control flow relationships among the basic blocks. We denote the $N \times N$ adjacent matrix as $A_{N \times N}$.

2) Syntax Feature Extraction: The syntax feature of a basic code block (i.e., a node in CFG) is extracted according to AST node types, variables, and function calls. In CFG, each basic block consists of one or more AST nodes combined together. There are approximately 140 distinct nodes in PHP syntax, which can be categorized into three main groups:

- The statement node, a language structure that does not produce a value and is not capable of occurring within an expression, such as a class definition;
- The expression node, a language structure that generates a value and can therefore be utilized within other expressions;
- The scalar node, which represents scalar values, like 'string' or magic constants, like '__FILE__'.
- In addition to these three groups there exists nodes that do not belong to either of these categories, such as names and call arguments.

Some special node types, parameters, and global variables appear frequently in webshells, which is an important symbol to identify these web scripts. Therefore, we use a GCN [12] combined with MinCutPool [13] pooling to implement the same functionality exhibit continuity, making the corresponding block features similar in the CFG. These blocks will be called as functional clusters in the following text. We can aggregate them by optimizing this problem, which is equivalent to maximizing:

$$
\maximize \frac{1}{K} \sum_{k=1}^{K} \sum_{i,j \in V_k} E_{i,j},
$$

where the numerator counts the number of edges in the same functional cluster and the denominator counts the number of edges between different functional clusters. Let $C \in \{0, 1\}^{N \times K}$ be a functional cluster assignment matrix, s.t. $C_{i,j} = 1$ if node $i$ belongs to functional cluster $j$. Then the minCUT problem can be expressed as:

$$
\maximize \frac{1}{K} \sum_{k=1}^{K} C_k^T A C_k,
$$

where $C_k$ is the $k$-th column of $C$ and $D$ is the degree matrix of $A$. For a given input $H^l$, the MinCutPool layer uses a multi-layer perceptron (MLP) with the softmax activation function to predict the functional cluster assignments:

$$
C = MLP(H^l, \theta_{MLP}),
$$

where $\theta_{MLP}$ is the parameters of MLP and optimized using the minCUT problem loss. According to the func-
IV. Experiments

A. Experimental Setup

In this section, we present and discuss our experimental results. Especially, we answer the following three research questions:

- **RQ1**: How effective is our graph representation for modeling PHP source code? (Section IV-B)
- **RQ2**: How well does our structural-aware model detect webshells? (Section IV-C)
- **RQ3**: If SAWD achieves better performance than state-of-the-art webshell detection systems? (Section IV-D)

1) **Data Preparation**: Currently, there is no standardized dataset available to evaluate the effectiveness of webshell detection systems. Thus, we manually collect webshells from 46 Github repositories and obtain normal samples from popular PHP projects with large user bases, such as Thinkphp and WordPress. We first remove duplicates from the collected webshells and normal samples via the md5 hash. Then, we classify the webshells according to our standards into three categories: big trojan (length greater than 2000 bytes), small trojan (length between 200 bytes and 2000 bytes), and one-word trojan (length less than 200 bytes). We obtain 911 big trojans, 1433 small trojans, and 407 one-word trojans. Finally, this collected webshell dataset is divided into a training dataset and a test dataset according to the proportion of 50% and 50%.

2) **Experimental Settings**: We train SAWD with an SGD optimizer for 200 epochs, and the learning rate is set to $5 \times e^{-4}$. The proposed approach is implemented using Python 3.8.0 and PyTorch 1.7.1.

B. Efficacy of Graph Representation (to RQ1)

To evaluate the efficacy of the graph representation for modeling PHP source code, we compare the performance of our keyword-based method with other syntax feature extraction methods. To implement the Word2Vec approach, we limit the feature extraction process to the first 40 tokens of each basic block. Besides, we assign a 150-dimensional vector for each token, resulting in a 6000-dimensional representation for each basic block. In the case of the Doc2Vec method, we extract features with a dimension of 1200 for each basic block, which is the same dimension as our keywords-based method. As Fig. 3 shows, the keywords-based method achieves an accuracy of 94.50% and outperforms Word2Vec and Doc2Vec. This benefits from its ability to selectively consider only the most informative words, thus retaining the original information of basic blocks to the maximum extent. In contrast, the Word2Vec method consumes excessive resources by storing all the features of each word in a block and must truncate the words, resulting in the loss of some key semantics. As for the Doc2Vec method, it considers the text of each block as a whole, which leads to the influence of many low-information words on feature extraction.

C. Detection Performance of Our Model (to RQ2)

To evaluate the detection performance of our structural-aware model, we generate AST from PHP source code and adopt CNN as the classifier. Note that we use the same top 1200 most frequent keywords as block features to ensure fairness. As shown in Fig. 4, we can observe that our model performs better than the CNN-based model. The reason is that our model can leverage GCN to combine graph representation for more efficient feature extraction. In addition, we can observe that the accuracy drops by about 0.5% after removing graph pooling, indicating that graph pooling indeed improves the generalization ability.

D. Comparison with Other Methods (to RQ3)

To demonstrate the effectiveness of our approach, we compare SAWD with a range of state-of-the-art webshell detection systems on our collected dataset. We use NeoPI [14] to extract multiple statistical features from the
source code files and use support vector machines (SVM), random forests (RF), and multi-layer perception (MLP) as the classifier. The opcode-based methods [15] adopt FastText to obtain the vectorized features of the opcode sequence, which is input into three machine learning algorithms for webshell detection. In addition, two DL-based methods proposed in [4] and [8] are also introduced for comparison.

We can observe from TABLE I that DL-based detection methods generally outperform traditional machine learning methods. This is because DL-based methods can automatically extract higher-level features from the data. Moreover, benefiting from SAWD preserving more complete syntax and semantic information, our method achieves better performance than other DL-based methods with an accuracy of 94.5%. Thus, it can be concluded that SAWD can effectively analyze PHP source code for webshell detection.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>$F_1$ Score</th>
<th>Miss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statics+SVM</td>
<td>61.5%</td>
<td>69.4%</td>
<td>8.7%</td>
</tr>
<tr>
<td>Statics+RF</td>
<td>90.9%</td>
<td>90.0%</td>
<td>14.2%</td>
</tr>
<tr>
<td>Statics+MLP</td>
<td>87.3%</td>
<td>87.2%</td>
<td>11.8%</td>
</tr>
<tr>
<td>Opcode+SVM</td>
<td>92.0%</td>
<td>91.4%</td>
<td>10.2%</td>
</tr>
<tr>
<td>Opcode+RF</td>
<td>92.3%</td>
<td>91.8%</td>
<td>9.7%</td>
</tr>
<tr>
<td>Opcode+MLP</td>
<td>92.5%</td>
<td>91.9%</td>
<td>10.8%</td>
</tr>
<tr>
<td>SC+W2V+GRU</td>
<td>93.8%</td>
<td>93.2%</td>
<td>4.7%</td>
</tr>
<tr>
<td>MSDetector</td>
<td>93.3%</td>
<td>92.7%</td>
<td>6.3%</td>
</tr>
<tr>
<td>SAWD (Ours)</td>
<td>94.5%</td>
<td>94.1%</td>
<td>3.4%</td>
</tr>
</tbody>
</table>

TABLE I. Comparison of metrics with other methods.

V. Conclusion

In this paper, we propose SAWD, a structural-aware webshell detection system, for PHP source code analysis. SAWD introduces an effective graph representation to preserve the syntax and semantic information in PHP source code. Based on the graph representation, SAWD leverages GCN combined with graph pooling for more efficient code analysis. To evaluate the performance of our proposed method, we conduct extensive experiments on the collected webshell dataset. The results demonstrate that our method outperforms state-of-the-art webshell detection systems. In future work, we will explore graph construction methods to replace CFG to further improve detection performance.

Acknowledgment

We would like to thank Robin Luo for his support and constructive comments on this work. This work was supported in part by SJTU-Q'ANXIN Joint Lab of Information System Security, NIO Student Innovation Funding Program, and Network Security & Ecosystem Innovation Laboratory of Intelligent Radio & Television Broadcasting of the National Radio and Television Administration under Grant TXX20200001ZSB001. Zhi Xue and Ruijie Zhao are the corresponding authors (e-mails: zxue, ruijiezhaoo@sjtu.edu.cn).

References

LTL \(_f\) Satisfiability Checking via Formula Progression

Tong Niu\(^1\), Yicong Xu\(^1\), Shengping Xiao\(^1\), Lili Xiao\(^2\), Yanhong Huang\(^1\), Jianwen Li\(^1\)
1. East China Normal University
2. Donghua University

Abstract

Linear Temporal Logic over finite traces, or LTL\(_f\), is a popular logic to describe specifications with finite behaviors in AI scenarios such as motion planning. Satisfiability is one of the fundamental problems of LTL\(_f\) and extensive studies have been conducted to speed up the process to check whether a given LTL\(_f\) formula is satisfiable. This paper presents a new approach, namely LSCFP, to solve the problem of LTL\(_f\) satisfiability checking by leveraging the formula progression technique. Compared to previous work, LSCFP utilizes formula progression to gather more information propagated along with the search path such that it can find satisfiable models more quickly if the input formula is satisfiable. A comprehensive experimental evaluation has been conducted to show the efficiency of LSCFP, and the results suggest that LSCFP is able to gain at least 15% performance improvement on checking satisfiable formulas when compared to the state-of-the-art LTL\(_f\) satisfiability checker aaltaf.

1. Introduction

Linear Temporal Logic over Finite Traces, abbreviated as LTL\(_f\), is a kind of logic that has emerged as a popular specification language in the AI domain to formalize and validate system behaviors [8]. Unlike standard Linear Temporal Logic (LTL) which is interpreted over infinite traces [16], LTL\(_f\) is interpreted over finite traces. While LTL is typically used in formal-verification settings, where we are interested in nonterminating computations, cf. [20], LTL\(_f\) is more attractive in AI scenarios focusing on finite behaviors, such as planning [15, 5] and user preferences [4, 19]. Due to the wide spectrum of applications of LTL\(_f\) in the AI community, the fundamental problems of LTL\(_f\), e.g., satisfiability [13, 12] and synthesis [11, 22], have been extensively studied in prior work. Towards applications, researchers successfully reduced the planning problem with LTL\(_f\) goals to the synthesis problem for LTL\(_f\) [7, 6, 1, 2, 21], which makes the logic very attractive in this domain.

This paper focuses on the problem of LTL\(_f\) satisfiability checking. Given an LTL\(_f\) formula, the satisfiability problem asks whether there is a finite trace that satisfies the formula. The existing solutions for LTL\(_f\) satisfiability checking fall into two categories. The first is to reduce the LTL\(_f\) satisfiability problem to that of LTL satisfiability, and then leverage the state-of-the-art LTL checkers, cf. [17], to solve LTL\(_f\) satisfiability. Thus, LTL\(_f\) satisfiability checking can benefit from progress in LTL satisfiability checking. There is, however, an inherent drawback that an extra cost has to be paid when checking LTL formulas, as the tool searches for a “lasso” (a lasso consists of a finite path plus a cycle, representing an infinite trace), whereas models of LTL\(_f\) formulas are just finite traces.

Based on this motivation, the second solution is proposed to reduce LTL\(_f\) satisfiability checking to the language emptiness check over the finite automaton that accepts the same languages as the input formula. Essentially, one can construct such an NFA (Non-deterministic Finite Automaton) from the given formula, such that the formula is satisfiable if and only if there is a finite trace that can be accepted by the NFA. [13] is the first attempt to implement this solution, and the subsequent work [14] improves the performance significantly by introducing the SAT technique to achieve the on-the-fly checking procedure, which is considered the most advanced approach so far.

This paper proposes to check the satisfiability of LTL\(_f\) formulas on the fly based on DFA (Deterministic Finite Automaton), which is constructed by leveraging the formula progression technique [3], and shows that this new approach, namely LSCFP, can potentially achieve a better performance, in particular on satisfiable formulas than that based on NFA [14]. Even though constructing DFA is a much harder task than constructing NFA from an LTL\(_f\) formula\(^1\), the DFA state contains more formula information than the NFA state such that a satisfiable model may be detected more quickly.

Take the formula \( \varphi = (a U b \land G \neg b) \lor a \) as an ex-

\(^1\)In theory, the translation to NFA is exponential blow-up while the translation to DFA is double exponential blow-up.
ample. The CDLSC approach, which is presented in [14], may first construct a successor \( \varphi_1 = a | b \land G \neg b \) of \( \varphi \) (\( \varphi_1 \) is an NFA state) and then determine \( \varphi_1 \) is unsatisfiable. Then the algorithm backtracks to \( \varphi \) and constructs another successor \( \varphi_2 = a \) of \( \varphi \), after which it can return satisfiable. The whole checking process may invoke at least two SAT calls. Meanwhile, the LSCFP approach presented in this paper uses formula progression to construct the successor \( \varphi_3 = (a | b \land G \neg b) \lor a \), which is a DFA state and can immediately be determined as an accepting state. Therefore, LSCFP is able to check the satisfiability of \( \varphi \) by invoking only one SAT call.

We compare LSCFP to CDLSC by conducting a comprehensive experiment on the widely used formulas for benchmarking LTL\(_f\) satisfiability. The results indeed affirm our conjecture that LSCFP can perform better than CDLSC on checking satisfiable formulas by achieving a 15% speed-up on average. Notably, LSCFP cannot compete to CDLSC on checking unsatisfiable formulas, the result of which is consistent with the fact that constructing DFA is much more costly than constructing NFA.

The rest of this paper is organized as follows. Section 2 introduces definitions for LTL\(_f\) and its satisfiability problem; Section 3 introduces the LSCFP approach in detail; Section 4 presents the experimental results, and finally, Section 5 concludes the paper.

2. Preliminaries

2.1. LTL over Finite Traces (LTL\(_f\))

Linear Temporal Logic over finite traces, or LTL\(_f\) [8], extends propositional logic with finite-horizon temporal connectives. In particular, LTL\(_f\) can be considered as a variant of Linear Temporal Logic (LTL) [16]. Distinguished with LTL, which is interpreted over infinite traces, LTL\(_f\) is interpreted over finite traces. Given a set of atomic propositions \( P \), the syntax of LTL\(_f\) is identical to LTL, and defined as:

\[
\varphi ::= tt \mid p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \varphi U \varphi
\]

where \( tt \) represents the true formula, \( p \in P \) is an atomic proposition, \( \neg \) is the negation, \( \land \) is the and, \( \lor \) is the strong Next and \( U \) is the Until operator. We also have the corresponding dual operators \( ff \) (false) for \( tt \), \( \lor \) (or) for \( \land \), \( \bullet \) (weak Next) for \( O \) and \( R \) (Release) for \( U \). Moreover, we use the notation \( G \varphi \) (Global) and \( F \varphi \) (Future) to represent \( ff \varphi \land \varphi \) and \( tt U \varphi \), respectively. Notably, \( O \) is the standard Next operator, while \( \bullet \) is weak Next; \( O \) requires the existence of a successor instance, while \( \bullet \) does not. Thus \( \bullet \) is always true in the last instance of a finite trace, since no successor exists there.

A finite trace \( \rho = \rho[0], \rho[1], \ldots, \rho[n] \) (\( n \geq 0 \)) is a sequence of propositional interpretations (sets), in which \( \rho[m] \in 2^P \) (\( 0 \leq m < |\rho| \)) is the \( m \)-th interpretation of \( \rho \), and \( |\rho| = n + 1 \) represents the length of \( \rho \). Intuitively, \( \rho[m] \) is interpreted as the set of propositions that are true at instant \( m \). We denote \( \rho_i \) to represent \( \rho[i], \rho[i+1], \ldots, \rho[n] \), which is the suffix of \( \rho \) from position \( i \).

LTL\(_f\) formulas are interpreted over finite traces. For a finite trace \( \rho \) and an LTL\(_f\) formula \( \varphi \), we define the satisfaction relation \( \rho \models \varphi \) (i.e., \( \rho \) is a model of \( \varphi \)) as follows:

- \( \rho \models tt \);
- \( \rho \models p \iff p \in \rho[0] \), where \( p \) is an atomic proposition;
- \( \rho \models \neg \varphi \iff \rho \not\models \varphi \);
- \( \rho \models \varphi_1 \land \varphi_2 \iff \rho \models \varphi_1 \) and \( \rho \models \varphi_2 \);
- \( \rho \models O \varphi \iff |\rho| > 1 \) and \( \rho_1 \models \varphi \);
- \( \rho \models \varphi_1 U \varphi_2 \iff \text{there exists } i \text{ with } 0 \leq i < |\rho| \text{ such that } \rho_i \models \varphi_2 \) and for every \( j \) with \( 0 \leq j < i \) it holds that \( \rho_j \models \varphi_1 \).

Two LTL\(_f\) formulas \( \varphi_1 \) and \( \varphi_2 \) are semantically equivalent, denoted as \( \varphi_1 \equiv \varphi_2 \), iff for every finite trace \( \rho, \rho \models \varphi_1 \) iff \( \rho \models \varphi_2 \). According to the semantics of LTL\(_f\) formulas, it is trivial to have that \( ff \equiv \neg tt \), \( O \phi \equiv \neg \bullet \neg \phi \), \( (\phi_1 U \phi_2) \equiv \neg(\neg \phi_1 \land \neg \phi_2) \) and \( G \phi \equiv \neg F \neg \phi \). A literal is an atom \( p \in P \) or its negation (\( \neg p \)). We say an LTL\(_f\) formula is in Negation Normal Form (NNF) if the negation operator appears only in front of an atom. Every LTL\(_f\) formula can be converted into its equivalent NNF in linear time. We now introduce below the concept of satisfiability for LTL\(_f\) formulas.

**Definition 1 (LTL\(_f\) Satisfiability).** An LTL\(_f\) formula \( \varphi \) is satisfiable iff there exists a finite trace \( \rho \in (2^P)^* \) such that \( \rho \models \varphi \); otherwise, it is unsatisfiable.

**Theorem 1 ([8]).** Checking the satisfiability of an LTL\(_f\) formula is PSPACE-complete.

**Notations.** We use \( cl(\phi) \) to denote the set of subformulas of \( \phi \). Let \( A \) be a set of LTL\(_f\) formulas, we denote \( \bigwedge_{\phi \in A} \psi \). We say an LTL\(_f\) formula \( \phi \) is in Tail Normal Form (TNF) if \( \phi \) is in Negated Normal Form (NNF) and \( \neg \)-free. Assume \( \phi \) is in NNF, \( trav(\phi) \) is defined as \( t(\phi) \land FTail \), where \( Tail \) is a new atom to identify the last state of satisfying traces (Motivated from [8]), and \( t(\phi) \) is an LTL\(_f\) formula defined recursively as follows:

1. \( t(\phi) = \phi \) if \( \phi \) is true, \( ff \) or a literal;
2. \( t(O \psi) = \neg Tail \land O(t(\psi)) \);
3. \( t(\bullet \psi) = Tail \lor O(t(\psi)) \);
4. \( t(\phi_1 \land \phi_2) = t(\phi_1) \land t(\phi_2) \);
5. \( t(\phi_1 \lor \phi_2) = t(\phi_1) \lor t(\phi_2) \);
6. \( t(\phi_1 U \phi_2) = (\neg T a i l \land t(\phi_1)) U t(\phi_2) \);
7. \( t(\phi_1 R \phi_2) = (T a i l \lor t(\phi_1)) R t(\phi_2) \).

**Theorem 2** ([12]). \( \phi \) is satisfiable iff \( \text{tnf}(\phi) \) is satisfiable.

In the rest of the paper, unless clearly specified, the input LTL\(_f\) formula is in TNF.

### 2.2. Transition-based Deterministic Finite Automaton (TDFA)

The Transition-based Deterministic Finite Automaton (TDFA) is a variant of the Deterministic Finite Automaton (DFA), which identifies the accepting condition on the transitions instead of states.

**Definition 2** (Transition-based DFA [18]). A transition-based DFA (TDFA) is a tuple \( A = (2^P, S, s_0, \delta, T) \) where

- \( 2^P \) is the alphabet;
- \( S \) is the set of states;
- \( s_0 \in S \) is the initial state;
- \( \delta : S \times 2^P \rightarrow S \) is the transition function;
- \( T \subseteq \delta \) is the set of accepting transitions.

For simplicity, we use the notation \( s_i \xrightarrow{\varphi} s_j \) to denote \( \delta(s_i, \varphi) = s_j \). The run \( r \) of a TDFA \( A \) on a finite trace \( \rho = \rho[0], \rho[1], \ldots, \rho[n] \in (2^P)^+ \) is a finite state sequence \( r = s_0, s_1, \ldots, s_n \) such that \( s_0 \) is the initial state, \( s_i \xrightarrow{\rho[i]} s_{i+1} \) is true for \( 0 \leq i < n \). Note that runs of TDFA do not need to include the destination state of the last transition, which is implicit \( s_{n+1} = \delta(s_n, \rho[n]) \), since the starting state (\( s_n \)) together with the labels of the transition (\( \rho[n] \)) are sufficient to determine the destination. \( r \) is called acyclic if \( (s_i = s_j) \Leftrightarrow (i = j) \) for \( 0 \leq i, j < n \). Also, we say that \( \rho \) runs across \( s_i \) if \( s_i \) is in the corresponding run \( r \). The trace \( \rho \) is accepted by \( A \) iff the corresponding run \( r \) ends with an accepting transition, i.e., \( \delta(s_n, \rho[n]) \in T \). The set of finite traces accepted by a TDFA \( A \) is the language of \( A \), denoted as \( \mathcal{L}(A) \).

According to [18], TDFA has the same expressiveness as the normal DFA, and for an LTL\(_f\) formula \( \varphi \), there is a TDFA \( A_\varphi \) such that \( \mathcal{L}(\varphi) = \mathcal{L}(A_\varphi) \). As a result, the LTL\(_f\) satisfiability-checking problem can be solved on the corresponding TDFA.

### 3. LTL\(_f\) Satisfiability Checking via Formula Progression (LSCFP)

In this section, we first introduce the concept of formula progression for LTL\(_f\) formulas and how to construct the TDFA via formula progression. Then we produce an on-the-fly satisfiability-checking framework along with the TDFA construction.

#### 3.1 LTL\(_f\)-to-TDFA via Formula Progression

The formula progression technique originates in [3] for goal planning with temporal logic. A definition of LTL\(_f\) progression has been used in [10], and here we adapt the definition to a finite trace instead of a single proposition.

**Definition 3** (Formula Progression for LTL\(_f\)). Given an LTL\(_f\) formula \( \varphi \) and a non-empty finite trace \( \rho \), the progression formula \( \text{fp}(\varphi, \rho) \) is recursively defined as follows:

- \( \text{fp}(tt, \rho) = tt \) and \( \text{fp}(ff, \rho) = ff \);  
- \( \text{fp}(p, \rho) = tt \) if \( p \in \rho[0] \); \( \text{fp}(p, \rho) = ff \) if \( p \notin \rho[0] \);  
- \( \text{fp}(\neg \varphi, \rho) = \neg \text{fp}(\varphi, \rho) \);  
- \( \text{fp}(\varphi_1 \land \varphi_2, \rho) = \text{fp}(\varphi_1, \rho) \land \text{fp}(\varphi_2, \rho) \);  
- \( \text{fp}(\varphi_1 \lor \varphi_2, \rho) = \text{fp}(\varphi_1, \rho) \lor \text{fp}(\varphi_2, \rho) \);  
- \( \text{fp}(\varnothing, \rho) = \varphi \) if \( |\rho| = 1 \); Else \( \text{fp}(\varnothing, \rho) = \text{fp}(\varphi, \rho_1) \);  
- \( \text{fp}(\bullet \varphi, \rho) = \varphi \) if \( |\rho| = 1 \); Else \( \text{fp}(\bullet \varphi, \rho) = \text{fp}(\varphi, \rho_1) \);  
- \( \text{fp}(\varphi_1 U \varphi_2, \rho) = \text{fp}(\varphi_2, \rho) \lor (\text{fp}(\varphi_1, \rho) \land \text{fp}(\varnothing U \varphi_2, \rho)) \);  
- \( \text{fp}(\varphi_1 R \varphi_2, \rho) = \text{fp}(\varphi_2, \rho) \land (\text{fp}(\varphi_1, \rho) \lor \text{fp}(\varnothing U \varphi_2, \rho)) \).

The following lemmas are not hard to obtain based on Definition 3, whose proofs are omitted here.

**Lemma 1.** Given an LTL\(_f\) formula \( \varphi \) and two non-empty finite traces \( \rho_1 \) and \( \rho_2 \), \( \rho_1 \models \text{fp}(\varphi, \rho_1) \) implies \( \rho_1 \cdot \rho_2 \models \varphi \).

**Lemma 2.** Given an LTL\(_f\) formula \( \varphi \) and two non-empty finite traces \( \rho_1 \) and \( \rho_2 \), it holds that \( \text{fp}(\text{fp}(\varphi, \rho_1), \rho_2) = \text{fp}(\varphi, \rho_1 \cdot \rho_2) \).

**Lemma 3.** Given an LTL\(_f\) formula and a non-empty finite trace \( \rho \), \( \rho \models \varphi \) implies \( \rho_i \models \text{fp}(\varphi, \rho^i) \) for every \( 0 \leq i < |\rho| \).

Now we re-construct the TDFA for an LTL\(_f\) formula.

**Definition 4** (LTL\(_f\) to TDFA). Given an LTL\(_f\) formula \( \varphi \), the TDFA \( A_\varphi \) is a tuple \((2^P, S, \delta, s_0, T)\) such that
• $2^P$ is the alphabet, where $P$ is the set of atoms of $\varphi$;

• $S = \{ \varphi \cup \text{fp}(\varphi, \rho) \mid \forall \rho \in (2^P)^+ \}$ is the set of states;

• $s_0 = \varphi$ is the initial state;

• $\delta : S \times 2^P \rightarrow S$ is the transition function such that $\delta(s, \sigma) = \text{fp}(s, \sigma)$ for $s \in S$ and $\sigma \in 2^P$ (Here $\sigma$ is considered a trace with length 1);

• $T = \{ s_1 \xrightarrow{\sigma} s_2 \in \delta \mid \sigma \models s_1 \}$ is the set of accepting transitions.

**Theorem 3.** Given an LTL$_f$ formula $\varphi$ and the TDFA $A_{\varphi}$ constructed by Definition 4, it holds that $L(\varphi) = L(A_{\varphi})$.

**Proof.** Let $|\rho| = n + 1$ ($n \geq 0$) and the corresponding run $r$ of $A_{\varphi}$ on $\rho$ is $s_0, s_1, \ldots, s_n$, where $s_0 = \varphi$.

$(\Leftarrow)$ According to Definition 4, $\rho$ is accepted by $A_{\varphi}$ implies $(\rho_0 = \rho[n]) \models s_n = \text{fp}(\varphi, \rho^n)$. Then from Lemma 1, we have $(\rho^n \cdot \rho_n = \rho) \models (s_n = \varphi)$.

$(\Rightarrow)$ First from Definition 4, every $\text{fp}(\varphi, \rho^i)$ for $0 \leq i \leq n$ is a state of $A_{\varphi}$. Secondly, $\delta(\text{fp}(\varphi, \rho^i), \rho[i]) = \text{fp}(\varphi, \rho^{i+1})$ is true for $0 \leq i \leq n$, because $\text{fp}(\varphi, \rho^{i+1}) = \text{fp}(\text{fp}(\varphi, \rho^i), \rho[i])$ is true (Lemma 2). Therefore, let $s_i = \text{fp}(\varphi, \rho^i)$ (0 $\leq i \leq n$) and the state sequence $r = s_0, s_1, \ldots, s_n$ is a run of $A_{\varphi}$ on $\rho$. Finally, $\rho \models \varphi$ implies that $\rho_n \models (s_n = \text{fp}(\varphi, \rho^n))$ is true because of Lemma 3. So $\rho$ is accepted by $A_{\varphi}$.

\[ \square \]

### 3.2 On-the-fly Checking Framework

The CDLSC approach [12] is an on-the-fly satisfiability checking framework along with the NFA construction. Compared to that, LSCFP completes the satisfiability checking over the on-the-fly DFA construction instead. In a high-level description, LSCFP combines SAT computation together with formula progression to construct DFA states instead of NFA ones.

Given an LTL$_f$ formula $\varphi$ whose closure is $cl(\varphi)$, every state $s$ of the corresponding DFA is a conjunction of subformulas of $\varphi$, i.e., $s \subseteq cl(\varphi)$. To compute one successor of $\varphi$, the SAT-based method proposed in [12] first converts $\varphi$ to its neXt Normal Form (XNF) $\text{xnf}(\varphi)$, in which there are no Until or Release subformulas of $\varphi$ in the atomic level or connected by Boolean operators. For example, $\varphi_2 = (\neg \text{Tail} \land a \land \text{Tail} \land a \land b)$ is not in XNF because there is an Until subformula ($\varphi$ itself) in the atomic level, while $\varphi_2 = (b \lor (\neg \text{Tail} \land a \land (\text{Tail} \land a \land b)))$ is, though $\varphi_1$ and $\varphi_2$ are semantically equivalent. In fact, every LTL$_f$ formula $\varphi$ has a linear-time conversion to an equivalent formula in XNF. By treating $\text{xnf}(\varphi)$ as a propositional logic, denoted as $\text{xnf}(\varphi)^P$, a Boolean SAT solver is able to return an assignment that indicates the current conditions (label of the NFA transition) and one next state of $\varphi$. Consider $\varphi_2$ as the example, the SAT solver may return $\{ a, b, \neg \text{Tail}, X((\neg \text{Tail} \land a) \land b) \}$ as an assignment, based on which CDLSC identifies that $\varphi_1 = (\neg \text{Tail} \land a) \land b$ is a successor of itself under the condition $a \land b \land \neg \text{Tail}$.

**Algorithm 1** LSCFP: LTL$_f$ Satisfiability Checking via Formula Progression.

**Required:** An LTL$_f$ formula $\varphi$.

**Ensure:** SAT or UNSAT.

1: if Tail $\land \text{xnf}(\varphi)^P$ is satisfiable then
2:    return SAT;
3: else
4:    Set $\psi = \text{xnf}(\varphi) \land \neg X(\varphi)$;
5:    while $\psi$ is satisfiable do
6:        Let $A$ be a propositional assignment for $\psi^P$;
7:        Let $L(A)$ be the set of literals extracted from $A$, i.e., $L(A)$ represents the current conditions;
8:        Let $\varphi' = \text{fp}(\psi \land \varphi, L(A))$, i.e., be the DFA successor state of $\varphi$ computed via formula progression;
9:        if LSCFP ($\varphi'$) returns SAT then
10:            return SAT;
11:        else
12:            Let $\psi = \text{xnf}(\varphi') \land \neg X(\varphi')$;
13:            return UNSAT;

Compared to CDLSC, LSCFP takes the current conditions as the input to compute $\text{fp}(\varphi, \{ a, b, \neg \text{Tail} \})$, which from Definition 3 equals $\varphi_1 \lor tt \equiv tt$. Therefore, LSCFP identifies $tt$ as a DFA successor state of $\varphi$ under the condition $a \land b \land \neg \text{Tail}$. In general, CDLSC computes only one successor under the fixed conditions, while LSCFP is able to compute all possible successors under the same conditions. Figure 1 shows the differences between LSCFP and CDLSC on constructing automata states.

The implementation of LSCFP is shown in Algorithm 1. Line 1 first checks whether the input formula $\varphi$ is a final state of the DFA, which is equivalent to checking whether $\text{Tail} \land (\text{xnf}(s))^P$ is satisfiable [12]. If $\varphi$ is not a final state,
the while loop (Line 4-10) searches the successors of $\varphi$ recursively to identify whether they can be final. Line 3 and 10 block the states already checked during the search. If no new state (assignment) can be generated from the SAT solver, the algorithm terminates with an unsatisfiable result.

In the while loop, Line 5 obtains the propositional assignment $A$ from the SAT solver, Line 6 extracts the conditions (label) of the current transition indicated by $A$, and then Line utilizes the formula progression technique to compute the corresponding TDFA successor.

The theorem below provides the theoretical guarantee to the correctness of Algorithm 1.

**Theorem 4.** The input $\text{LTL}_f$ formula is satisfiable if and only if Algorithm 1 returns SAT.

**Proof.** First of all, it is not hard to see that Algorithm 1 constructs exactly the TDFA in Definition 4. Secondly, let $\rho = \omega_0\omega_1\ldots\omega_n$ ($n \geq 0$) be a non-empty trace and the corresponding run on the TDFA constructed in Algorithm 1 be $r = s_0(\varphi) \xrightarrow{\omega_0} s_1 \ldots s_n \xrightarrow{\omega_n} s_{n+1}$. Notably, since Algorithm 1 constructs TDFA, the run of the automaton on a given trace is unique. According to Theorem 3, $\rho \models \varphi$ if and only if $\omega_n \models s_n$ holds, which indicates that $s_n$ is a final state. Then, $s_n$ is a final state if and only if Algorithm 1 can return SAT at Line 2. As a result, $\rho \models \varphi$ if and only if Algorithm 1 returns SAT at Line 2 when taking $s_n$ as the input in the recursive checking process.

4. Experimental Evaluations

**Experimental Setup.** We implement LSCFP in the state-of-the-art LTL$_f$ satisfiability solver aaltaf$^2$, which implemented CDLSC, and use Minisat 2.2.0 [9] as the SAT engine. To evaluate the performance of LSCFP, we compared it to CDLSC on the widely used LTL$_f$ benchmarks that are presented in [12], in which there are 8645 formulas in total.

We ran the experiments on a RedHat 6.0 cluster with 2304 processor cores in 192 nodes (12 processor cores per node), running at 2.83 GHz with 48GB of RAM per node. When we ran the experiments, each tool was run on a dedicated node, which guarantees that no CPU or memory conflict with other jobs can occur. For each tested formula, we set the timeout to 120 seconds, measuring execution time with Unix time. Excluding timeouts, the results from both LSCFP and CDLSC are consistent.

**Results.** Figure 2 shows the comparison between LSCFP and CDLSC on satisfiable formulas in terms of time cost. LSCFP performs better than CDLSC on more than 60% of the tested satisfiable cases, and for some particular cases, LSCFP can be 10X times faster than CDLSC. LSCFP is able to solve 6380 satisfiable instances within timeout, while CDLSC solves the number of 6419. Notably, there are 39 formulas that cannot be solved by LSCFP within timeout, which slows down its overall performance. However, LSCFP is still able to obtain a 15% improvement on average when compared to CDLSC on checking satisfiable formulas.

Figure 3 shows the comparison between LSCFP and CDLSC on unsatisfiable formulas in terms of time cost. For instances that can be solved by both approaches, LSCFP gains a similar performance with CDLSC. However, LSCFP has more timeout cases than CDLSC. In total, LSCFP solves the amount of 1904 unsatisfiable formulas, while CDLSC solves the amount of 2141. So there are 237 more unsatisfiable instances that can be timeout for LSCFP than CDLSC, which is the main reason why LSCFP is not competitive in solving unsatisfiable formulas. In our conjecture, LSCFP needs to enumerate all TDFA states in the worst case to check unsatisfiability, and this is a much

---

$^2$https://github.com/lijwen2748/aaltaf
heavier task than enumerating NFA states. Also, CDLSC leverages more advanced convergent techniques by using the conflict sequence, which we plan to explore in LSCFP to speed up the performance.

5 Concluding Remarks

In this paper, we present LSCFP, a new on-the-fly satisfiability checking approach for LTL formulas. Compared to the existing CDLSC method, LSCFP combines SAT invoking together with the formula progression technique to compute the DFA states instead of NFA ones. The benefit is to accelerate the checking performance, in particular for satisfiable instances. Our experimental results affirm our conjecture on such a benefit. Currently, LSCFP cannot perform well on checking unsatisfiable formulas, and we plan to introduce the ideas in CDLSC to enable fast convergent by overapproximating reachable states in future work.

6 Acknowledgment

We thank anonymous reviewers for their helpful comments. Tong Niu and Yicong Xu have equal contributions to this paper. Jianwen Li is the corresponding author. This work is supported by the National Natural Science Foundation of China (Grant #U21B2015 and #62002118), the Shanghai Collaborative Innovation Center of Trusted Industry Internet Software, and the National Key Research and Development Program (2022YFB3305202).

References

Formal Specification and Verification of an Autonomous Vehicle Control System by the OTS/CafeOBJ method

Yifan Wang Masaki Nakamura Kazutoshi Sakakibara Yuki Okura
Toyama Prefectural University, Toyama, Japan

Abstract- The autonomous vehicle control system is a typical kind of hybrid system that combines both continuous and discrete behavior. Formal specification and verification techniques help us to verify desired properties of given systems. In this study, we propose a way to describe a formal specification of an autonomous vehicle control system in CafeOBJ algebraic specification language. The control system is a hybrid system with continuous variables of time, velocity, and position controlled by discrete pedal actions including acceleration, braking, and no-operation. We also verify the safety property of the autonomous vehicle control system by a theorem proving technique called the proof score method.

Keywords: Autonomous vehicle; Hybrid system; Formal verification; Observational transition system; Proof score method

I. Introduction

A hybrid system is a dynamic system that includes both continuous and discrete dynamic behavior. The autonomous vehicle has both continuous behaviors (velocity and position) and discrete behaviors (pedal actions), therefore, in order to design and verify an autonomous vehicle control system we model it as a hybrid system. There are many methods used to test and simulate autonomous vehicles, such as CarMaker † and SUMO ‡, but only testing and simulation just provide some kinds of limited or predetermined paths, it is far from enough for safety. We need formal specification and verification techniques to make sure safety of the autonomous vehicle control system.

Formal verification is an approach to verify that a given specification satisfies some desired properties formally. One approach of formal verification is model checking. In our previous work [1], we proposed a way to describe and verify an autonomous vehicle group control system by Maude model checker and showed Maude is useful to design the system with the safety property. Although model checking is fully automated, the state space is limited where we choose a time sampling strategy instead of dense time.

The other approach of formal verification is theorem proving where mathematical proofs are made by the interaction of humans and computers, it is semi-automated but applicable to infinite state space and it supports continuous variables. CafeOBJ supports specification execution based on a rewrite theory for theorem proving. The OTS/CafeOBJ method is a formal method in which a system is modeled as an observational transition system (OTS), its specification is described in CafeOBJ, and properties are verified formally based on the specification execution, called the proof score method [2].

There have been several case studies of the OTS/CafeOBJ method which deal with systems with only discrete behaviors ([2] and so on). In [3], the OTS/CafeOBJ method is applied to distributed real-time systems, which are kinds of hybrid systems with only one continuous variable of time. In [4], the OTS/CafeOBJ method is applied to multitask hybrid systems, which includes only a simple hybrid system for explanation’s sake. In [5], the safety property of an autonomous vehicle intersection traffic control system by the OTS/CafeOBJ method is shown, where the system includes only one continuous variable of real-time.

In this study, we propose a way to describe a formal specification of an autonomous vehicle control system in CafeOBJ algebraic specification language. The control system is a hybrid system with continuous variables of time, velocity, and position controlled by discrete pedal actions including acceleration, braking, and no-operation. We also verify the safety property of the autonomous vehicle control system by a theorem proving technique called the proof score method.

II. Hybrid Automaton

Hybrid automata are models of hybrid systems which contain the discrete and continuous behavior and we give a model of an autonomous vehicle control system as a hybrid system according to the literature [6]. In this article, we consider a hybrid automaton of a single autonomous vehicle control system, represented in Figure 1. The system consists of a single autonomous vehicle with three locations:
acceleration, nothing, and brake. The pedal states of acceleration, brake, and no-operation (nothing) can be selected freely.

The operational semantics of hybrid automata is given as a state transition system where a state is represented by a tuple of a location and values of variables and there are discrete jump transitions between locations and continuous flow transitions for time elapsing (See [1] for more details). The safety property of our system is given as follows formally: for any reachable state, the position of the vehicle is less than forty-five, that is the obstacle position.

III. OTS/CafeOBJ specification

In this section, we introduce a way to describe the OTS/CafeOBJ specifications of a hybrid system. We model an autonomous vehicle control system with a single vehicle as hybrid automata and describe them as OTS/CafeOBJ specifications. Then we give the simulation of our system and certify that the vehicle will stop before the obstacle.

A CafeOBJ specification consists of modules, in which sorts, operators, and equations are declared in module LABEL, module RAT, and module VEHICLE. A module is declared with mod or just mod. The name of a module is written after mod. Module elements are declared between {}.

CafeOBJ modules can be classified into tight modules and loose modules. Tight and loose modules begin with mod! and mod* respectively. A loose module mod* denotes all models satisfying axioms. A tight module mod! denotes the initial model.

A Data modules

An OTS/CafeOBJ specification consists of data modules and a system module. We first give a data module LABEL for the vehicle.

The name of the module is LABEL. The module declaration with mod! denotes the tight denotation, where the module denotes only the initial model. In the initial model, any elements of a carrier set are represented by a term constructed from its signature, and no two elements of a carrier set are equivalent unless the corresponding terms can be shown to be equal using its axioms.

We set a tight data module LABEL specifying three constant operators accel, nothing, and brake of Label, a binary predicate =, and a variable is declared with the keyword var. And we define the equality predicate, which takes two labels and returns true if they are the same, otherwise false.
A system module is given as a behavioral specification of CafeOBJ. A behavioral specification has a special sort, called a hidden sort, and special operations called behavioral operations, whose arguments include the hidden sort. A behavioral operation whose returned sort is not hidden is called an observation and whose returned sort is hidden is called a transition. Two elements of the hidden sort are observationally equivalent if their observed values are equivalent for each observation. An OTS/CafeOBJ specification is a restricted behavioral specification, where observational equivalence is preserved by transitions. The following is an OTS/CafeOBJ specification of an autonomous vehicle system:

\[
\text{eq loc1(a(S))} = \text{accel} \quad \text{eq loc1(b(S))} = \text{brake} \quad \text{eq loc1(n(S))} = \text{nothing}
\]

Next, we specify the continuous transition. Time advancing \(\text{tick}\) is described as follows:

\[
\text{ceq now(tick(T,S))} = \text{now}(S) + T \quad \text{if} \quad \text{c-tick}(T,S). \\
\text{ceq loc1(tick(T,S))} = \text{loc1}(S) \quad \text{if} \quad \text{c-tick}(T,S).
\]

The first equation specifies the updated value of \(\text{now}\) is set to \(T\) time later, that is, \(\text{now}^+ = \text{now} + T\) if the effective condition is satisfied. The second equation specifies the variable \(\text{loc}\) is unchanged if it satisfies the \(\text{c-tick}\).

In the OTS/CafeOBJ method, we describe an equation like \(x(tick(T,S)) = \text{rhs} (\text{right-hand side})\) if \(\text{c-tick}(T,S)\). \(x\) is the observation for a continuous variable. The term \(\text{tick}(T,S)\) stands for the result state by applying \(\text{tick}(T,\_\_\_)\) to the state \(S\), that is, the state after time advancing by \(T\) from \(S\). The left-hand side \(x(tick(T,S))\) of the equation stands for the value of \(x\) for the state \(tick(T,S)\). By this equation, the value of observation \(\text{tick}(T,S))\) is defined as the right-hand side when \(\text{c-tick}(T,S)\) is true. The value of \(x\) after \(T\) is obtained from the flow condition of the hybrid automaton.

Let \(v1(t)\) be the value of \(v1\) at time \(t\). From the flow condition \(v'_1 = a\) where \(a\) is a constant \((a = -1, 0, 1)\), the value \(v1(T)\) of \(v1\) at the state after \(T\) is calculated as follows: \(v1(T) = v1(0) + \int_0^T adt = v1(0) + [at]_0^T = v1(0) + a * T\) where the value of \(v1(0)\) is the value of \(v1\) at \(S\). Thus, the right-hand side of the equation whose left-hand side is \(v1(tick(T,S))\) can be written as \(v1(S) + a(\text{loc1}(S)) * T\), where \(a(\text{accel}) = 1, a(\text{brake}) = -1\) and \(a(\text{nothing}) = 0\). To describe the equation, we introduce an operation \(\text{nextv}(V,A,T)\) which calculates the value of \(v1\) after \(T\) from the state whose value of \(v1\) is \(V\) and acceleration value is \(A\).

\[
\text{eq nextv}(V,A,T) = V + A * T \\
\text{ceq v1(tick(T,S))} = \text{nextv}(v1(S), a(\text{loc1}(S)), T) \quad \text{if} \quad \text{c-tick}(T,S)
\]

Similarly, \(x1(t)\) is the value of \(x1\) at time \(t\). From the flow condition \(x'_1 = v\), the value \(x1(T)\) of \(x1\) at the state after \(T\) is calculated as follows: \(x1(T) = x1(0) + \int_0^T v(t)dt = x1(0) + [v(T)]_0^T = x1(0) + v + [v]_0^T + T = x1(0) + v_0 * T + T / 2 + a(T) * T^2\), where the value of \(x1(0)\) is the value of \(x1\) at \(S\). Thus, the right-hand side of the equation whose left-hand side is \(x1(tick(T,S))\) can be written as \(x0 + v0 * T + T / 2 + a1(\text{loc1}(S)) * T^2\). To describe the equation, we introduce an operation \(\text{nextx}(X,V,A,T)\) which calculates the value of \(x1\) after \(T\) from the state whose value of \(x1\) is \(X\) and acceleration value is \(A\).

\[
\text{eq nextx}(X,V,A,T) = X + T / 2 * A + T * T + X
\]
The above conditional equations have \( c\text{-}\text{tick}(T, S) \) as their conditions. The effective condition \( c\text{-}\text{tick}(T, S) \) is given by invariants of the hybrid automaton which \( \text{eq} \ c\text{-}\text{tick}(T, S) = \text{inv}_b \text{ and } \text{inv}_a \text{ and } \text{inv}_n. \)

Invariants in hybrid automata should always hold, which means that time cannot advance if the invariants do not hold. Thus, the effective condition \( c\text{-}\text{tick} \) is given as the conjunction of all invariants. For example, the \( \text{inv}_b \) equals \((\text{loc1}(S) = \text{brake} \implies (0 \leq \text{nextv}(v_1(S), a_1(\text{loc1}(S)), T) \text{ and } \text{nextv}(v_1(S), a_1(\text{loc1}(S)), T) \leq 4)))\) means that \(0 \leq \text{nextv} \leq 4\) holds after \(\text{tick}_x\) whenever the location is the braking state. The effective condition of \(\text{tick}_x\) should check invariant \(\text{loc1}(S)\) by the updated value of \(\text{nextv}\) and \(\text{nextx}\). In other words, if future values violate invariants, time cannot advance.

One of our future works is to apply the proposed method to practical applications of multitask hybrid systems with multiple autonomous vehicles. There are several related work on formal verification for hybrid system, e.g. SpaceEx\(^3\) and KeYmaera X\(^3\). Another one of our future work is to compare them with our approach and combine them to obtain more efficient formal verification.

**References**


---

\(^3\)https://spaceex.imag.fr/

\(^4\)https://keymaerax.org/publications.html

---

IV. Verification of hybrid system

A single reduction can prove a simple equation. More complex properties are proved by combining several reductions, which is called a proof score. First, we give a state predicate \(\text{inv1}(S)\) such that the vehicle does not exist over the position of 45 at the state \(S\).

\[
\text{eq } \text{inv1}(S) = (x_1(S) < 45) .
\]

If we prove \(\text{inv1}(S)\) for all reachable states from the initial state and get the result is true, the safety property holds. As the induction basis, we apply the reduction command to \(\text{inv1}(\text{init})\) to prove the initial state to satisfy \(\text{inv1}\).

open \text{INV} .
red \text{inv1}(\text{init}) .
close

CafeOBJ interpreter returns true, which implies the induction basis holds, that is, \(\text{inv1}\) holds at the initial state.

To complete the proof, we need to prove not only the safety property \(\text{inv1}\) but also a lemma \(\text{inv2}\). We make two lemmas and 12 proof passages, all of which return true. Because of the page limitation, we omit the induction part of \(\text{inv1}\), an introduction of a lemma \(\text{inv2}\) and its proof. The codes of them can be found in our GitHub page \(^4\).

V. CONCLUSION

We described an observational transition system of an autonomous vehicle control system as an example of a hybrid system, and verified the safety property by the proof score method.

---

\(^4\)https://github.com/evan-jaapan/SEKE2023.git
Assistance in the management of rule sets for rule-based expert systems

Magali Legast and Axel Legay
UCLouvain
Louvain-la-Neuve, Belgium
{magali.legast, axel.legay}@uclouvain.be

Abstract—Rule-based expert systems (RBES) use knowledge about a specific topic, represented as rules, in order to solve particular problems that would otherwise require a human expert. The creation and maintenance of the rule sets used come with several challenges in order to guarantee that they remain free of any error, which could reduce performances or lead to erroneous results.

In this paper, we present a methodology to provide an automated assistance for domain experts creating and maintaining rules for RBES. This assistance takes the form of automated detection of relationships between rules that can lead to redundancies or conflicts. By reducing the weight borne by the human experts in the verification of the rules, it reduces the chance of errors, which helps increase the relevance of the rule set.

To complete the theoretical methodology, we have implemented a functional prototype allowing for the management of a rule set and the visual highlight of redundancies and potential conflicts.

Our approach, developed in the context of a case study, can be used for rules in any domain as long as they can be described in the same format. The approach can also be extended or modified to account for other types of relationships between rules.

Index Terms—rule-based expert system, rule set

I. INTRODUCTION

Expert systems (ES) are computer applications that contain knowledge about a specific topic with which it solves particular problems that would otherwise require a human expert. Using this knowledge, it can reach a conclusion that can be given to the user [2–4], [6], [8], [11], [12].

There exist many kinds of ES, considering eleven categories. Amongst them are fuzzy expert systems, neural networks, case-based reasoning and many others. In rule-based expert systems (RBES), the information is represented as a set of IF-THEN rules with an antecedent and a consequent. The rules are then used to perform operations on data, make inferences and reach a conclusion [3], [8], [11], [12]. Such systems are used in a wide variety of applications, such as disease diagnosis, energy optimization or firewalls. Many other examples of rule-based ES can be found in [13].

Since the result provided by the system depends on the knowledge encoded in its rules, its quality is highly dependent on the rules themselves. In many cases, those rules are written by domain experts. It can be a challenge for them to guarantee that the rule set is and remains free of errors that can at best reduce the performance of the ES or at worst lead to an erroneous result, as is further explained in Section II.

This work tackles the difficulty for humans to write and maintain a good rule set. Our contribution is an approach to assist them in the management of the sets through an automated detection of redundancies or potential conflicts. To do so, we have developed an efficient methodology to identify the different types of relationships between pairs of rules, using matrices and numerical representations. We have built on [5], which shows how to detect anomalies in firewall rule sets, to adapt that work for the identification of other types of relationships. The methodology is detailed in Section III.

We have also implemented a functional prototype, presented in Section IV, to demonstrate how our methodology could be implemented in practice. This prototype allows to create and maintain rule sets, using the assistance provided by the automatic identification of relationships between rules. By alleviating the weight of such verification, it can help domain experts to obtain more accurate rules.

We have applied this problem to a case study, described in Section V. Nonetheless, our approach is not limited to a specific field and can even be extended to identify other types of relationships depending on the needs of the application, as is discussed in Section V.

II. RULE-BASED EXPERT SYSTEMS FOR ENERGY OPTIMIZATION - A CASE STUDY

This research was motivated by a case study aiming at improving an RBES that provides recommendations to optimize energy production and consumption for WeSmart [16]. This company supports energy communities, which are communities in which participants can produce and consume energy together in a more autonomous and efficient way.

The rule-based system considered in this case study has a knowledge base with rules that link conditions on input energy data to an appropriate recommendation. Their construction is similar to many other rules, including those in firewalls [17], [13], [15]. They can be viewed as having the form

IF ⟨condition⟩ THEN ⟨recommendation⟩

The condition is a set of attributes and their associated values that define a specific set of situations. It can be seen as a boolean expression that is True when the situation considered is in the set described by the condition and False otherwise. When a situation satisfies the condition of a certain rule, those situation and rule are said to match. The recommendations

DOI Reference Number: 10.18293/SEKE2023-189
are pieces of advice that can be given to a user in the specific situations described by the rule condition. An example of such rules is given in Table I with the recommendation and the attributes of the condition. An unspecified value indicates the absence of constraint for that attribute and can be represented with a wildcard ‘∗’, as in [7]. For example, the 5th rule in Table I corresponds to IF (YearlyOfftake ∈ [350, ∞], MaxProd = False) THEN (Recommendation = “Add solar panels”).

The RBES checks the condition of each rule in the set against the situation given as input. It then gives as output all the recommendations associated with the matching rules. Several different recommendations can thus be aggregated in the same output. The ordering of the rules is not significant. The implementation of such an ES is trivial, we have thus not developed it further.

The relevance and accuracy of those rules is crucial since they have a direct impact on how useful the output recommendations will be. Like in many other applications, they are created manually by experts in the field. Those experts can face some challenges in order to guarantee that the rule set allows the ES to have the desired behavior. Besides the correctness of each rule, they also need to avoid unwanted contradictions or redundancies within the rule set, i.e. distinct rules that contain conflicting or identical information. This can be a quite tedious and error-prone task while the number of rules grows, as reported by WeSmart experts. This is thus a difficulty at the creation of the rule set, but also for its maintenance and the addition of new rules as the system and the knowledge evolve.

Since the rule creation and maintenance process is both of high importance and high error risk, the proposed solution is to assist the human experts in this task. This assistance takes the form of an automatic verification of the possibility of conflict or redundancy between each two rules within the set. The verification methodology is described in Section III. Identifying rules that can lead to these problems helps experts easily detect and fix a rule that would not yield the wanted result or would bring undesired redundancy. It thus reduces the chances of mistakes and helps with the obtaining of a more relevant rule set and the overall improvement of the RBES.

It is worth noting that some level of redundancy may be useful to make rules more understandable or reduce their number. Additionally, different recommendations may be complementary to one another, like “Add photovoltaic panels” and “Reduce use of energy at night”, while others may be conflicting, like “Run highly consuming devices at night” and “Reduce use of energy at night”. For these reasons, WeSmart requested for domain experts to have a strong manual control on the rules. So it was desired to have an automated detection of potential problems, but no automated correction.

### III. Identification of Relationships Between Rules

The unwanted behaviors mentioned in Section II can happen when the same situation matches two different rules. In such cases, there is redundancy if the two rules have the same recommendation and contradiction if their recommendations are conflicting. In order to detect those cases, we need to look at the relationships between each pair of rules. The relationship between two rules is considered regarding the relationships between the set of situations that can be matched by each of those rules. This is similar to the way firewall rules are considered regarding the set of packets they match.

In this section, we will define the different types of relationships considered for the case study in Subsection III-A describe the matrices and numerical representations used to represent them in Subsection III-B then describe in Subsection III-C how we can use those representations to detect connections. Finally, we show how to identify each type of connection in Subsection III-D and how the numerical encoding needs to be defined in Subsection III-E. Our contribution uses the work done on the relationships between firewalls rules in [5] and adapts it to other applications.

#### A. Relationship definitions

Two rules can either be disconnected, if there can be no situation they both match, or connected, if there exists at least one possible situation they can both match. The relationships between two rules, inspired by [1, 5], are defined below:

- **Disjunction/Disconnection:** Two rules $r$ and $s$ are disjoint if the set of situations that are matched by both rules is empty. The values of at least one of their respective attributes are disjoint.

- **Equality:** Two rules $r$ and $s$ are equal if all the situations matched by $r$ are also matched by $s$ and all the situations matched by $s$ are also matched by $r$. The values of all of their respective attributes are equal.
c) **Inclusion:** A rule \( r \) is included in a rule \( s \) if all the situations matched by \( r \) are also matched by \( s \) and there exist situations that are matched by \( s \) but not by \( r \). For all respective attributes, \( r \)'s values are either a subset of or equal to \( s \)'s values, with at least one attribute for which it is a subset.

d) **Overlap:** Two rules \( r \) and \( s \) overlap if there can exist at least one situation that is matched by \( r \) and not by \( s \), at least one situation that is matched by \( s \) and not by \( r \) and at least one situation that is matched by both \( r \) and \( s \). For all respective attributes, the values of \( r \) and the values of \( s \) can’t be disjoint and at least one of the two following sufficient conditions must hold:

- There is at least one attribute for which the values of \( r \) are a subset of the values of \( s \) and at least one attribute for which the values of \( s \) are a subset of the values of \( r \).
- There is at least one attribute for which the values of \( r \) overlap the values of \( s \), meaning the intersection between the two set of values is not empty, not equal to the set of values of \( r \) and not equal to the set of values of \( s \).

Equality, inclusion and overlap are different types of connection, while disjunction is the only type of disconnection. In addition to those relationships, two rules can also have the same or different recommendations.

Reference [5] considers those relationships (although named differently), with a different overlap definition that doesn’t consider overlaps between attribute values. It also defines anomalies specific to firewall rules sets and develops the detection methodology for those anomalies. On the other hand, our approach directly considers the relationships, which are more general and can be further specified by indicating if the recommendations are equal or different for both rules. We thus adapt the approach in [5] to identify them.

**B. Matrix representation and Inter-Difference Coding**

We propose to represent a set of rules in a matrix \( S \) where the rows represent the \( n \) rules and the columns the \( m + 1 \) fields. Those fields are the \( m \) attributes, preceded by the recommendation which is located in the first field. The element \( v_{ij} \) thus represents the value of the \( j \)th field for the \( i \)th rule.

Since a relationship between rules can be defined with regard to the relationships between their respective fields, the latter are represented in Inter-Difference Matrices (IDM). For each attribute, there is a corresponding IDM \( R \) that represents the relationships between the values of this attribute for each pair of rules. There is also one IDM for the recommendations. For a rule set of \( n \) rules with \( m \) attributes, there is thus \( m + 1 \) IDM’s of size \( n \times n \). Since the relationship between attribute values of rules \( R_i \) and \( R_j \) are reciprocal, \( R \) is a strictly upper triangular matrix. The information for a relationship between rules can be found in the entry \( (i, j) \) if \( i < j \) and in the entry \( (j, i) \) if \( j < i \).

Together, the IDM matrices can be considered as several layers of a 3D tensor, creating an IDM layers model. The \( 0 \)th layer is associated to the recommendation. The other layers, \( 1 \)st to \( m \)th, correspond to each of the attributes. We can define an Inter-Difference Vector (IDV) \( R_{ij} \) that represents the relationship between the rules \( R_i \) and \( R_j \), where \( i < j \). Its elements are the elements \( (i, j) \) of each IDM, which includes the recommendation IDM. Its length is thus \((m + 1)\). A visual representation of the IDM layer model and an example of vector \( R_{ij} \) can be seen in Figure 1.

The concepts of IDM, IDM layers model and IDV have been introduced in [5]. They have been adapted here, primarily to include the recommendations in the IDM layers model and in the IDV, which simplifies the mathematical representation of relationships, allowing the use of a unique vector to do so. Another change is the use of triangular matrices to avoid storing redundant data.

![Image](image.png)

**Fig. 1. IDM layers model, illustration taken from [5]**

In the IDM’s, the relationships between the respective attribute values or recommendations of two rules are encoded using a specific Inter-Difference Coding (IDC). The IDC associates a numerical value to each type of relationship, which will be used in the identification, as exposed in the following subsections. The IDC used for the case study is given in Table II alongside the definitions of the different possible relationships between respective fields. \( \mathbb{R}(v_{ik}, v_{jk}) \) corresponds to the relationship between \( R_i \) and \( R_j \) for the \( k \)th field, which is represented in the \( k \)th field of the IDM layers model. Two unspecified values are considered to be equal, and a specified value to be a subset of an unspecified value. The use of an IDC was introduced in [5]. We have renamed the inclusions and added an extra value for the overlap.

**TABLE II**

<table>
<thead>
<tr>
<th>( R_{ij} )</th>
<th>Definition (relationships between attributes)</th>
<th>IDC code</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{R}(v_{ik}, v_{jk}) ) ( 1 \leq k \leq m )</td>
<td>val((v_{ik}) \cap val(v_{jk}) = \emptyset )</td>
<td>0</td>
</tr>
<tr>
<td>Disjunction</td>
<td>val((v_{ik}) = val(v_{jk}))</td>
<td>1</td>
</tr>
<tr>
<td>Equality</td>
<td>val((v_{ik}) \subset val(v_{jk}))</td>
<td>2</td>
</tr>
<tr>
<td>Inclusion ij</td>
<td>val((v_{ik}) \supset val(v_{jk}))</td>
<td>3</td>
</tr>
<tr>
<td>Inclusion ji</td>
<td>val((v_{ik}) \cap val(v_{jk}) \neq \emptyset )</td>
<td>6</td>
</tr>
<tr>
<td>Overlap</td>
<td>val((v_{ik}) \not\subset val(v_{jk}))</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>val((v_{ik}) \not\supset val(v_{jk}))</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( R_{ij} )</th>
<th>Definition (relationships between recommendations)</th>
<th>IDC code</th>
</tr>
</thead>
<tbody>
<tr>
<td>k = 0</td>
<td>val((v_{ik}) \neq val(v_{jk}))</td>
<td>-1</td>
</tr>
<tr>
<td>Difference</td>
<td>val((v_{ik}) = val(v_{jk}))</td>
<td>1</td>
</tr>
<tr>
<td>Equality</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**C. Relationships between rules with regard to IDC**

Thanks to the IDC in Table II, the relationship between rules are defined with regards to \( R_{ij} \), adapting the work in [5] to
the relationships defined above and the use of a unique vector to represent them. Regarding the conditions of the rules and their attributes, the relationships are defined as:

- **Disjunction**: \( x \in \{1, \ldots, m\}, \ R_{ij}(x) = 0 \)
- **Equality**: \( \forall x \in \{1, \ldots, m\}, \ R_{ij}(x) = 1 \)
- **Inclusion of** \( R_i \) **in** \( R_j \):
  \[
  \exists x \in \{1, \ldots, m\}, \ R_{ij}(x) = 2
  \]
  \[
  \forall e \in (\{1, \ldots, m\}\ \backslash \{x\}), \ R_{ij}(e) \in \{1, 2\}
  \]
- **Inclusion of** \( R_j \) **in** \( R_i \):
  \[
  \exists x \in \{1, \ldots, m\}, \ R_{ij}(x) = 3
  \]
  \[
  \forall e \in (\{1, \ldots, m\}\ \backslash \{x\}), \ R_{ij}(e) \in \{1, 3\}
  \]
- **Overlap**:

  Either one of these conditions needs to be satisfied:

  \[
  \begin{cases}
  \exists x \in \{1, \ldots, m\}, \ R_{ij}(x) = 6 \\
  \forall e \in (\{1, \ldots, m\}\ \backslash \{x\}), \ R_{ij}(e) \in \{1, 2, 3, 6\}
  \end{cases}
  \]

  or

  \[
  \begin{cases}
  \exists x \in \{1, \ldots, m\}, \ R_{ij}(x) = 2 \\
  \exists y \in \{1, \ldots, m\}, \ R_{ij}(y) = 3 \\
  \forall e \in (\{1, \ldots, m\}\ \backslash \{x, y\}), \ R_{ij}(e) \in \{1, 2, 3, 6\}
  \end{cases}
  \]

To further specify the relationships between rules with regard to their recommendations, we have:

- **Same recommendation**: \( R_{ij}(0) = 1 \)
- **Different recommendations**: \( R_{ij}(0) = -1 \)

Those definitions allow to prove the existence of a connection between the rules \( R_i \) and \( R_j \), with \( i < j \), using the product \( p_{ij} \) of the elements of the IDV \( R_{ij} \) of the rules, defined as:

\[
p_{ij} = \prod_{x=0}^{m} R_{ij}(x)
\]

**Theorem 1**: A connection exists between rules \( R_i \) and \( R_j \), with \( i < j \), if and only if \( p_{ij} \neq 0 \).

The proof, which follows a similar reasoning as in [5], is trivial and can be found in [10].

**D. Identification of relationships using** \( p_{ij} \)

The product \( p_{ij} \) not only allows to detect the existence of a connection, but also to identify the type of the relationship. Indeed, thanks to the definitions of Section [II-C], we can express \( p_{ij} \) using the corresponding IDC values. The identification is then done using simple mathematical operations, such as modulo. Those \( p_{ij} \) values are stored in the Product Matrix \( P \) that is generated from the \((m+1)\) IDM’s, with \( P[i, j] = p_{ij} \) for every \( i < j \). This takes inspiration from [5], while adapting the representations and conditions to the relationships defined in Section [III-A].

The expression of \( p_{ij} \) with regard to the IDC values is given below for a selection of relationships taken as examples, as well as the condition on \( P[i, j] \) that allows to identify them.

- **Disjunction**:
  - \( p_{ij} = 0 \)
  - Condition: \( P[i, j] = 0 \)

- **Inclusion of** \( R_i \) **in** \( R_j \), same recommendation:
  \[
  P[i, j] \mod 6 \neq 0
  \]
  \[
  \begin{cases}
  P[i, j] \mod 2 = 0 \\
  P[i, j] > 0
  \end{cases}
  \]

- **Overlap, different recommendations**:
  \[
  P_{ij} = -1 \times (1^{(m-x-y-z)} \times 2^x \times 3^y \times 6^z)
  \]
  \[
  \begin{cases}
  P[i, j] \mod 6 = 0 \\
  P[i, j] < 0
  \end{cases}
  \]

with \( m \) the total number of attributes, \( x \) the number inclusions \( ij \) amongst the attributes, \( y \) the number of an inclusions \( ji \) and \( z \) the number of overlaps. The full list of expressions and conditions is available in [10].

**E. Adaptation of IDC to relationships**

The values in the IDC that represent the relationships between attribute values need to be chosen in order to facilitate the expression and identification of relationships between rules using simple mathematical operations.

The values for the difference between attribute values has to be 0, the absorbing element of multiplication, in order for the theorem to be true.

When a relationship between rules can be deduced from the presence of one kind of relationship between attributes, like for equality or inclusion, its IDC value has to be a prime number to allow the identification using modulo operations. The choice of 1, the identity element of multiplication, for the equality simplifies the identification conditions.

When a relationship between rules can be the result of different relationships between attributes, like for the overlap, taking it into consideration in the IDC code simplifies the identification condition. Hence, the IDC value for the overlap is the product of the values for inclusion \( ji \) and inclusion \( ij \).

The values for the difference and equality between recommendations of course need to be of opposite sign, an absolute value of 1 being the obvious choice to simplify the identification conditions.

Following these principles, other IDC values can be chosen to represent other kinds of relationships.

**IV. RELATIONSHIP IDENTIFICATION TOOL**

The methodology has been implemented in a functional prototype, which demonstrates how the theoretical methodology exposed in Section [III] can be implemented and used in practice. Its code is available at [9], alongside rule set examples, including those in Tables [II] and [III]. It has been written with Python 3.7 and is distributed under MIT license.

The Relationship Identification Tool (RIT) allows the user, supposedly a domain expert, to manage a rule set through a graphical user interface (GUI), which can be seen in Figure [2] It highlights and indicates the different types of connections between rules, using the methodology described previously. It also supports the addition, modification and deletion of rules or attributes in order to create and maintain the rule set. The user
can thus successively analyze them and their relationships, then modify the rule set until the intended behavior is obtained.

We tested the tool with a simplified and modified version of WeSmart rule set\(^1\) which contain interesting test examples. We started with the rule set presented in Table I which is composed of 9 rules, thus involving 36 relationships. With the tool, we can select each rule to view the relationships it has with each of the other rules, as is done for rule 6 in Figure 2. For the complete rule set, there is 23 overlaps and 8 cover inclusions. We detected that 5 of these overlaps and 6 of these inclusions involved conflicting recommendations, thus bringing unwanted behavior. To fix this rule set, we deleted rule 5 which was conflicting with 4 others (rules 0, 1, 2 with inclusions and rule 8 with overlap). We corrected attribute values for rules 3 and 4 to remove their overlap. This was also done for the 3 first rules so they wouldn’t be included within each other any more and to avoid the overlap with rule 8. As a result, there remain the 8 rules (24 relationships) listed in Table III out of which the 5 firsts had some attribute values updated. Between those rules, there are 18 overlaps and 2 inclusions, all desired in order to provide complementary recommendations when applicable. More detail about the correction process in relation with the methodology can be found in [10].

While we tested the tool with the energy rules of the case study, it can be used for rules of any domain. Its implementation can also be easily extended to account for other relationship types between rules. Further, the RuleSet class, which allows for the management of the ruleset objects, is independent from the GUI and can thus be reused as is in other implementations.

V. DISCUSSION AND FUTURE WORK

The methodology presented provides an efficient way to assist in the management of a rule set and the relationships between its rules. For a new set, the time complexity to identify all the relationships between pairs of rules is quadratic in the number of rules and linear in the number of attributes. The matrices used also allow to store those relationships. They can thus be updated in linear time when a rule is added or modified, which is particularly interesting for the maintenance of the rule set. New attributes can be added with the update of the matrices in quadratic time in the number of rules. All those complexities simply follow from the way the different matrices are built.

Our methodology, while developed for energy rules in a specific context, is generic and can be used as is to identify the relationships defined in Section III-A for rules of any domain. Furthermore, our solution and the relationships considered, which already adapt [5] that was designed for firewall rules, can also be modified and extended to identify other types of relationships depending on the need of a particular application. Indeed, other relationships between attributes can be represented with the IDC and lead to the identification of new types of relationships between rules, following the same process.

Beyond that extension, it would also be interesting to study the possibility to consider related attributes. Indeed, the methodology considers all attributes independently from each other. They can however be linked, like the average daily energy consumption and the average consumption over a certain period of time during the day. Such situation can lead to more connections between rules being detected than if those relationships between distinct attributes had been taken into account. This is thus a current limitation of the tool. Considering how the attributes may interact would give another insight into the relationships between the rules and give more precise results.

Another interesting enhancement would be to provide users with recommendation on potential correction, in addition to the display of relationships type. For example, the system could give more information on the potential conflicts between rules. With the current solution, the user has to determine whether the recommendations of two connected rules are

\(^1\)For confidentiality reasons, the actual rules and recommendations used in WeSmart system cannot be presented here. Therefore, all examples presented in this article have been created by the authors for illustration purpose. They remain plausible and useful to the discussion.
complementary or in conflict. It would thus be useful to identify recommendations that are always in conflict with one another and recommendations that are always complementary. This information could then be included in the feedback given to the user and would provide an extra assistance compared to the current solution.

It would also be useful to build on this model to add an automated correction of some or all conflicts and redundancies, depending on the specificity of the domain of interest and the RBES in which the rule set is used.

Lastly, more experiments on larger and more diverse rule sets would be interesting to better quantify the impact of our approach. Key points to study include the measured improvement of accuracy of the rule set and its RBES, the number of rules in the sets and the gain in time and quality of experience for the domain experts during the creation and maintenance process.

VI. Conclusion

In this work, we have considered a case study to improve an existing RBES used by a company in the field of energy communities. One of the key elements for the accuracy and efficiency of such systems is the quality of the set of rules in which its knowledge is contained. The creation and maintenance of those rules, typically handled by domain experts, can be a tedious process with high chances of errors.

In response to this challenge, we developed a methodology to assist domain experts in the management of such rule sets. This assistance takes the form of a tool that automatically verifies if there is a possibility of conflict or redundancy between each two rules within the set. By highlighting rules that can lead to these problems, it helps experts easily detect and fix a rule that would not yield the wanted result or that brings undesired redundancy. By reducing the weight of the verification borne by the domain experts, it reduces the chances of mistakes and helps with the attainment of a more relevant rule set and with the improvement of the RBES.

The proposed methodology relies on matrices and basic mathematical operations, which makes it efficient and easy to implement. While it has been developed for a specific case study, it can be applied to any rule set with the same redundancy and conflict concerns. It can also be easily adapted and extended to detect other types of relationships between rules.

Beyond the theoretical presentation of the methodology, we have implemented and tested it with a functional prototype which code is available at [9].

References


Modeling and Verification of Autonomous Driving Systems under Stochastic Spatio-Temporal Constraints

Mengyuan Wang¹, Tengfei Li², Jing Liu¹*, Hui Dou¹, Hongtao Chen⁴, John Zhang⁴*, Lipeng Zhang³*,
¹Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
²Casco signal Ltd., Shanghai, China
³ECNU-CASCO Trustworthy Railway Joint Lab, East China Normal University, Shanghai, China
⁴Huawei Technology, Shanghai, China
*Corresponding authors: Jing Liu (Email: jliu@sei.ecnu.edu.cn), John Zhang (Email: yuhongzhang@xidian.edu.cn), Lipeng Zhang (Email: 52194501021@stu.ecnu.edu.cn)

Abstract—The decision-making process in autonomous driving systems encounters large uncertainties with environmental changes and needs to face the complex spatio-temporal evolution of multiple objectives. Formal analysis and verification are crucial to establishing reliable and safe standards. In this paper, we propose an extension of the clock constraint language CCSL to construct spatio-temporal constraint and autonomous driving safety specifications, leveraging various autonomous driving scenarios. Additionally, we introduce probabilistic spatio-temporal events and devise extensions for driving specifications that incorporate stochasticity. This specification is converted to the UPPAAL-SMC model for facilitating formal modeling and verification. Specific schemes and verification are given in conjunction with a typical autonomous driving scenario.

Keywords—CCSL, uncertainty modeling, autonomous driving control, statistical model checking

I. INTRODUCTION

Cyber-Physical Systems (CPS) [1] are multi-dimensional and intricate systems that integrate the physical, network, and computing environments. As previously discussed, CPS is a combination of cyber and physical elements, which gives rise to various types of uncertainties. Autonomous Driving Systems (ADS) exemplify a typical instance of CPS. The uncertainties of human behaviors and the physical environment usually result in unavoidable stochastic behaviors of ADS.

Compared with manual driving, the accident rate of autonomous driving is lower. However, due to the complexity of the driving environment, improving the safety of the autonomous driving system is still a hot spot and a difficult area of research. Driving decisions based on rules or based on data in different scenarios should have different response strategies as an intelligent body. Unlike human drivers, autonomous vehicles must timely and accurately respond to the complex and dynamic environment, adhering to spatial and clock-related restrictions. In other words, the trigger conditions in spatio-temporal systems are constrained not only by strictly temporal limitations and physical time intervals but also by logical and spatial relationships.

MARTE [2] extends UML by providing comprehensive support for dense and discrete time, chronometric and logical time, as well as simple and multiple time references. As the companion language of MARTE, CCSL [3] enables the specification of clock mutual dependence. Thus, as a high-level formalism in the Formal Specification Level, CCSL accurately models the causal and temporal timing behaviors of real-time embedded systems.

In this paper, we focus on the spatio-temporal probability constraint logic and use this spatio-temporal event containing uncertainty for driving decisions.

This paper makes the following contributions:

• We propose a stochastic extension of CCSL with probability clock and stochastic delay to support modeling uncertainty-aware timing behaviors.
• We represent spatial events as multiform logical clocks of stochastic CCSL, called Stochastic stCCSL.
• We propose an encoding in SHA of the semantics of Stochastic stCCSL. Then we can check the safety specifications with the statistical model checker UPPAAL-SMC.

The rest of this paper is structured as follows: In Section II, we introduce preliminaries. Section III presents our extension of CCSL and proposes some transformation rules to encode Stochastic stCCSL into SHA. Section IV presents our case study and evaluation results of various solutions with UPPAAL-SMC. In Section V, we summarize related work and conclude in Section VI.

II. PRELIMINARIES

A. Spatial Relationship

Based on the topological space, RCC8 spatial relations are proposed. The main binary relations are shown in Fig. 1:

- $DC(AB)$: A and B are independent of each other, i.e., none of the points in A are in B, and vice versa.
- $EC(AB)$: A and B boundaries are tangent to each other, i.e., A and B just intersect.
PO(AB): A and B partially overlap, i.e., some points in A are in B, and vice versa.

TPP(AB): A is contained in B, the points in A are in B, and the boundary of A is tangent to B.

NTPP(AB): A is contained in B, and the points in A are completely in B.

EQ(AB): All points in A are contained in B and vice versa.

B. Clock Constraint Specification Language (CCSL)

Definition 1 (Logical Clock): A logical clock c is defined as an infinite sequence of ticks: \( (c_i)_{i=1}^{\infty} \).

Definition 2 (Schedule): A schedule is a function \( \sigma : \mathbb{N} \rightarrow 2^C \), where C is a set of logical clocks. Given an execution step \( s \in \mathbb{N} \), \( \sigma(s) \) denotes the set of clocks that tick at step s.

Definition 3 (History): Given a schedule \( \sigma : \mathbb{N} \rightarrow 2^C \), a history of a schedule \( \sigma \) is a function \( H_\sigma : C \times \mathbb{N}^+ \rightarrow \mathbb{N}^+ \):

\[
H_\sigma(c, n) = \begin{cases} 
0 & \text{if } n = 1 \\
H_\sigma(c, n - 1) & \text{if } n > 1 \land c \notin \sigma(n-1) \\
H_\sigma(c, n - 1) + 1 & \text{if } n > 1 \land c \in \sigma(n-1) 
\end{cases}
\]

Intuitively, \( H_\sigma(c, n) \) counts the number of c ticks before moment n.

Table I presents the syntax and semantics of CCSL operators. The semantics of CCSL is defined by the satisfaction of a schedule against corresponding constraints. Due to the page limit, we do not provide full details of the formal semantics of other CCSL constraints. Please refer to [3] for more details.

C. Stochastic Hybrid Automata (SHA)

Stochastic Hybrid Automata (SHA) [4] is described by a tuple \( H = (L, l_0, C, Act, I, F, pE) \), where: 1) L is a finite set of locations, 2) \( l_0 \) is the initial location, 3) C is a finite set of clocks, Act is the set of actions, 4) \( I : L \rightarrow Zones(C) \) assigns an invariant to each location, where Zones(C) is the set of zone in C, 5) F is a time delay function for each location, 6) \( pE \subseteq L \times Act \times Zones(C) \times prob \times 2^C \times \mathbb{L} \) is a finite set of transactions with probability, where prob \( \in [0, 1] \) is a rational number presenting the probability.

III. STOCHASTIC EXTENSION OF SPATIO-TEMPORAL CCSL

A. Spatio-temporal constraints

Spatio-temporal constraints events are proposed based on RCC8 spatial relations and spatial logical S4u. The spatial relations are shown in Fig. 1:

Spatial events [5] are derived from the evolution of spatial relationships. CCSL combining time and space constraints introduce the definition of the spatial events, which are generated by the interaction between areas varying through time.

Definition 4 (Spatial Event): The syntax of spatial events is defined as \( \mathcal{E}_{act} \), where \( act = \{ join(A, B), detach(A, B), include(A, B), exclude(A, B) \} \). Specific semantics of spatial events are as follows:

- \( \mathcal{E}_{join}(A, B) \) is used to express the transition from relation DC(A, B) to EC(A, B) \( \lor \) PO(A, B) \( \lor \) TPP(A, B) \( \lor \) NTPP(A, B).
- \( \mathcal{E}_{detach}(A, B) \) is used to express the transition from relation EC(A, B) \( \lor \) PO(A, B) \( \lor \) TPP(A, B) \( \lor \) NTPP(A, B) to DC(A, B).
- \( \mathcal{E}_{include}(A, B) \) is used to express the transition from relation DC(A, B) \( \lor \) EC(A, B) \( \lor \) PO(A, B) \( \lor \) TPP(A, B) \( \lor \) NTPP(A, B) to TPP(A, B) \( \lor \) NTPP(A, B).
- \( \mathcal{E}_{exclude}(A, B) \) is used to express the transition from relation TPP(A, B) \( \lor \) NTPP(A, B) to DC(A, B) \( \lor \) EC(A, B) \( \lor \) PO(A, B) \( \lor \) TPP(A, B) \( \lor \) NTPP(A, B).

The driving of autonomous vehicles involves spatio-temporal evolution, where their spatial position changes over time. Therefore, the driving protocol requires both logical time constraints to limit the system’s response migration time and spatial event constraints. The CCSL provides a logical clock with strong expressive and reasoning abilities.

Fig. 3 shows where \( Ego \), an autonomous vehicle, typically follows vehicle \( Preceding \). However, there is a potential risk of rear-end collisions when \( Ego \) is driving at high speed. The autonomous vehicle in this scenario must determine the logical state of the spatio-temporal relationship between the two vehicles and modify its driving behavior accordingly.

The algorithm of generating Spatial Event \( \mathcal{E}_{detach}(A, B) \) is shown in Algorithm 1.

In this function, obstacles encompass either vehicles or other obstacles. If there is a vehicle in the same lane as the autonomous vehicle, the function calculates the braking distance (brakingD) and the distance that the front vehicle can cover within the autonomous vehicle’s braking time (obstacleD). On the other hand, if there is an obstacle in the same lane, the function only considers the autonomous vehicle’s speed for

Fig. 2: Example for spatial relations evolving over time

Fig. 3: A scenario in straight lane
TABLE I: The syntax and semantics of CCSL

<table>
<thead>
<tr>
<th>Name</th>
<th>Constraint</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precedence</td>
<td>$a[d] &lt; b$</td>
<td>$\forall n \in \mathbb{N}^+. (H_\sigma(b,n) - H_\sigma(a,n) = d) \Rightarrow (b \notin \sigma(n))$</td>
</tr>
<tr>
<td>Causality</td>
<td>$a \leq b$</td>
<td>$\forall n \in \mathbb{N}^+. H_\sigma(a,n) \geq H_\sigma(b,n)$</td>
</tr>
<tr>
<td>Subclock</td>
<td>$a \subseteq b$</td>
<td>$\forall n \in \mathbb{N}^+. (a \in \sigma(n)) \Rightarrow (b \in \sigma(n))$</td>
</tr>
<tr>
<td>Exclusion</td>
<td>$a # b$</td>
<td>$\forall n \in \mathbb{N}^+. (a \notin \sigma(n)) \vee (b \notin \sigma(n))$</td>
</tr>
<tr>
<td>Union</td>
<td>$a \triangleleft b + c$</td>
<td>$\forall n \in \mathbb{N}^+. (a \in \sigma(n)) \Rightarrow (b \in \sigma(n) \vee c \in \sigma(n))$</td>
</tr>
<tr>
<td>Intersection</td>
<td>$a \triangleleft b \cdot c$</td>
<td>$\forall n \in \mathbb{N}^+. (a \in \sigma(n)) \Rightarrow (b \in \sigma(n)) \wedge (c \in \sigma(n))$</td>
</tr>
<tr>
<td>Infinum</td>
<td>$a \triangleleft b \cdot c$</td>
<td>$\forall n \in \mathbb{N}^+. H_\sigma(a,n) = \max (H_\sigma(b,n), H_\sigma(c,n))$</td>
</tr>
<tr>
<td>Supremum</td>
<td>$a \triangleleft b \vee c$</td>
<td>$\forall n \in \mathbb{N}^+. H_\sigma(a,n) = \min (H_\sigma(b,n), H_\sigma(c,n))$</td>
</tr>
<tr>
<td>Delay</td>
<td>$a \triangleleft b \cdot d$</td>
<td>$\forall n \in \mathbb{N}^+. H_\sigma(a,n) = \max (H_\sigma(b,n) - d, 0)$</td>
</tr>
<tr>
<td>DelayFor</td>
<td>$a \triangleleft b \cdot d$ on $c$</td>
<td>$\forall n \in \mathbb{N}^+. (a \in \sigma(n)) \Rightarrow (b \in \sigma(n) \land \exists m \in \mathbb{N}^+. (b \in \sigma(m) \land H'_\nu(c,n,m) = d))$</td>
</tr>
<tr>
<td>Periodicity</td>
<td>$a \triangleleft b \propto p$</td>
<td>$\forall n \in \mathbb{N}^+. (a \in \sigma(n)) \Rightarrow (b \in \sigma(n) \land (H_\sigma(b,n) + 1) \mod p = 0)$</td>
</tr>
</tbody>
</table>

Algorithm 1: Generate Spatial Event \(E_{\text{detach}}(A,B)\)

Input:
- Ensemble of obstacles, \(O(i), i \in n\);
- The gap between the autonomous vehicles, \(gap\);
- The current lane by the autonomous vehicle, \(V_{\text{lane}}\);
- The position of the autonomous vehicle \(V_{\text{pos}}\);

Output:
- True or False;
- Extracting the current lane and pos of obstacles(i);

while \(i < n\) do
  if \(V_{\text{lane}} == O(i).\text{lane}\) then
    if \(gap <= g_{\text{safe}}\) then
      \(V_{\text{new}} = V_{\text{pos}} + \text{brakingD}(v\text{Max})\);
      \(O_{\text{new}} = O(i)\.\text{pos} + \text{obstacleD}(v\text{Max})\);
      if \(O_{\text{new}} - V_{\text{new}} >= g_{\text{safe}}\) && \(O(i)\.\text{static}\) then return True;
    else if \(O(i)\.\text{pos} - V_{\text{new}} >= g_{\text{safe}}\) && \(O(i)\.\text{static}\) then return True;
  return False;

where parameter \(\text{lower}\), \(\text{upper} (\text{lower} < \text{upper})\) are two natural numbers representing the lower and upper delay bounds.

**Definition 7** (Stochastic DelayFor):

\[ a \triangleleft b \propto \text{delay}(F) \text{ on } c \] (2)

where delay function \(F\) is defined as two types of probability density function: \(\text{Normal}(\mu, \sigma), \text{Exp}(\theta)\). \(\text{delay}(F)\) describes the probability distribution of the waiting period before the timeout is reached.

Semantics of Stochastic stCCSL

To conduct thorough analyzes on CCSL specifications, [6] propose to represent the semantics using transition systems.

**Definition 8** (Labeled Transition System): Labeled Transition System (LTS) is defined as a tuple \(\mathcal{A} = (S, s_0, T, A)\), where: 1) \(S\) is a set of states, 2) \(s_0 \in S\) is the initial state, 3) \(T\) is a set of labels, 4) \(T \subseteq S \times A \times S\) is a set of transitions.

**Definition 9** (Clock-Labeled Transition System): Clock-Labeled Transition System (CLTS) is defined as a tuple \(\mathcal{A} = (S, s_0, T, C)\), where: 1) \(S\) is a set of states, 2) \(s_0 \in S\) is the initial state, 3) \(C\) is a finite set of clocks, 4) \(T \subseteq S \times 2^C \times \mathbb{R}^+\) is a set of transitions.

**Definition 10** (Stochastic Clock-Labeled Transition System): Stochastic Clock-Labeled Transition System (SCLTS) is defined as a tuple \(\mathcal{A} = (S, s_0, T, C, P, d, T)\), where: 1) \(S\) is a set of states, 2) \(s_0 \in S\) is the initial state, 3) \(C\) is a finite set of clocks, 4) \(P \subseteq Q\) is the set of rational numbers between 0 and 1, 5) \(d\) is a stochastic variant that follows the exponential distribution, 6) \(T \subseteq S \times 2^C \times P \times S\) is a set of transitions.

For instance, the constraint Subclock \(a \subseteq b\), its transition system is given in Fig. 4. The constraint Delay \(a \triangleleft b \cdot d\), its transition system is given in Fig. 5.

C. Transform Stochastic stCCSL to SHA

Some previous work [7] has considered the encoding of Mode/State-based MARTE/CCSL behavior into Timed Automata (TA). In this section, we incorporate stochastic and continuous behavior into SHA to reinforce it. In general, the mapping rules are summarized in Table II.
Meanwhile, CCSL utilizes logical clocks to define partial orders and causal relationships among events. In this study, we incorporate logical clocks to capture spatial and temporal constraints during the specification process. Specifically, the formulation of the spatial events and temporal constraints is achieved through the Stochastic stCCSL.

TABLE II: The mapping rules between Stochastic stCCSL and SHA.

<table>
<thead>
<tr>
<th>Stochastic stCCSL</th>
<th>SHA</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 \delta c_1h )</td>
<td>left</td>
<td>It denotes that the time delay in state ( c_1 ) is ( t )</td>
</tr>
<tr>
<td>( c_2 \delta c_1delay(Exp(rate)) )</td>
<td>exp</td>
<td>It denotes exponential time delay with parameter ( rate )</td>
</tr>
<tr>
<td>( c_2 \delta c_1[\mu,T,T'] )</td>
<td>unif</td>
<td>It denotes uniform distribution interval time delay</td>
</tr>
<tr>
<td>( c_2 \delta c_1delay(Normal(\mu,\sigma)) )</td>
<td>norm</td>
<td>It denotes Gaussian distribution time delay</td>
</tr>
<tr>
<td>( c_2(p_1)p_1\Rightarrow(p_2) )</td>
<td>prob</td>
<td>It denotes probabilistic choice</td>
</tr>
</tbody>
</table>

IV. CASE STUDY

A. A right turn scenario for Autonomous Vehicles

The perception devices in autonomous vehicles acquire environmental information from the surroundings, transmitting it to the recognizer. The recognizer identifies various elements such as traffic signs, obstacles, pedestrians, and surrounding vehicles, and makes judgments. These judgment results are transmitted to the controller, which generates corresponding control actions.

Fig. 6 shows a scenario for an autonomous vehicle taking a right turn at the intersection. In this scenario, the gray vehicle in the straight line is a human-driven car, and the green vehicle is an autonomous vehicle. Note that according to traffic rules, green Vehicle has a higher priority.

First of all, we give several requirements utilizing natural language. The requirements are as follows:

- **a)**: The autonomous vehicle probabilistically executes a right turn considering the time delay associated with environmental information.

- **b)**: Upon receiving the check command, the autonomous vehicle’s sensors collect environmental information every 50 ms with a jitter.

- **c)**: The classifier computes the spatial relationship between the autonomous vehicle and the vehicles in the through lane based on the collected environmental information, assuming the merging process concludes within \([25, 30]\) ms.

- **d)**: Area A represents the autonomous vehicle area, area B represents the through lane area, and area C represents the merging area. A must merge into and leave the C area before B can enter.

- **e)**: If A has not completely left the C area when B is about to enter the area, A needs to accelerate through the C area.

- **f)**: If the classifier predicts that both A and B will enter the C area simultaneously, A will not turn right and will decelerate until it stops.

- **g)**: The classifier sends the classification result to the controller within 30 ms.

- **h)**: The three requirements of \( d, e, \) and \( f \) for right turns are mutually exclusive, although execution can also be probabilistic.

B. Build the Stochastic stCCSL and map to UPPAAL-SMC model

We describe the requirement as the Stochastic stCCSL. Table III presents the specifications \( a–h \) and their corresponding verification results obtained after performing 1000 simulation runs.

Specification \( a \) involves receiving a right turn signal (TurnR) and an environment detection command. Specification \( b \) indicates a sensor’s completion of environmental data collection, introducing a delay centered around 50ms with a Gaussian distribution (mean \( \mu = 50 \), deviation \( \sigma = 5 \)). The specification of \( c \) calculates spatial relationships within a 25-30ms timeframe. The specification \( d-f \) indicates spatio-temporal constraint. Based on the calculated spatial event logical relationship function, \( d \) indicates that turning is safe, that is, \( exclude(A,C) \prec join(B,C) \), where \( exclude(A,C) \) and \( join(B,C) \) are obtained from the aforementioned spatio-temporal function calculation. If this spatio-temporal event logical relationship holds, the detector issues a \( turn_safe \)
signal. Similarly, if the $e$ logical relationship is satisfied, the detector issues a turn_risksignal. If $f$ is satisfied, the detector issues a turn_danger signal. $g$ indicates that these signals are sent to the controller within 30ms. $h$ expresses the mutual exclusion relationship of the first three spatial events.

C. Build the UPPAAL-SMC model for the whole system

Based on the establishment of Stochastic stCCSL and the verification of the foundation discussed earlier, Fig. 8 shows the overall UPPAAL model of the right turn scenario. The system calculates the transition to three states: ok_turn, risk_turn, and slowdown, based on the spatio-temporal evolution relationship. The vehicle continues normal operation during the transition from the ok_turn state to the normal state. When a vehicle transitions to the risk_turn state, it signifies a potential rear-end collision with the following vehicle. Consequently, the vehicle must accelerate to exit the risk_turn state and return to the normal state. If the vehicle enters the slowdown state, it slows down while receiving the periodic signal check. The transition of these three parallel states is currently based on logical spatio-temporal constraints. It is essential to emphasize the probabilistic nature of right turns, which are influenced by various environmental factors. Therefore, assessing the risk associated with right turns and choosing between normal, aggressive, or impossible options entails probabilistic decision-making in an uncertain environment.

![Image](image_url)

Fig. 7: Probability distribution with confidence 95% and 98% in the query

In order to verify the probability distribution of the time interval from when the system sends a right-turn signal to when it makes a decision, we illustrate a query of the model.

$Pr[\leq 100](<> \text{Process.decide})$

The query means the probability distribution of reaching state decide within 100-time units

We conducted two trials for this run using various statistical settings: 1) We set the statistical parameters of UPPAAL-SMC with $\alpha = 0.05, \epsilon = 0.04$. By simulating 10000 runs, the quantitative result is shown in Fig. 7(a). We can get a confidence interval $[61.8, 99.3]$ with a confidence 95%. 2) We set the statistical parameters of UPPAAL-SMC with $\alpha = 0.02, \epsilon = 0.01$. By simulating 10000 runs, the quantitative result is shown in Fig. 7(b). We can get a confidence interval $[59.2, 99.8]$ with a confidence 98%. In these figs, the x-axis indicates the time limit, and the y-axis denotes the probability density distribution.

V. RELATED WORK

This section compares our approach to related works. Du et al. [8] proposed pCCSL, a stochastic extension to MARTE/CCSL for modeling uncertainty in CPS, and used SMC to explore alternative solutions and drive the refinement process. They illustrate their proposition by modeling an energy-aware building. Huang et al. [9] proposed an extension of PrCCSL, called PrCCSL*, to specify and verify dynamic and stochastic behaviors for automotive systems using UPPAAL-SMC. Gao et al. [10] enhanced CCSL by adding parameters to constraints in order to represent uncertainties in temporal behaviors. Compared to their approach, our approach considers both spatio-temporal constraints and stochastic behavior simultaneously.

VI. CONCLUSION AND FUTURE WORK

In this paper, CCSDL is expanded to propose time delay constraint relations including probability intervals and density functions. We introduce time-like temporal constraints and develop logical relation functions for autonomous driving. A mapping method is proposed for this expanded constraint language, facilitating the conversion and verification using the UPPAAL model. This spatio-temporal probabilistic language is applied to the right turn example, involving formal modeling, distribution verification, and overall model evaluation for the system. These advancements enable more precise modeling of uncertainties in intelligent vehicle systems.

In our future work, we propose integrating deep learning-based uncertainty models with rule-based mathematical models to construct comprehensive traffic regulations for typical autonomous driving scenarios. Additionally, we plan to develop refined models to enhance evaluation and validation.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development under Project 2022YFB3305202, the NSFC under Project 61972150, Shanghai Trusted Industry Internet Software Collaborative Innovation Center, and Shanghai Post-Doctoral Excellence Program 2021146.

REFERENCES

### TABLE III: Verification Results

<table>
<thead>
<tr>
<th>Req</th>
<th>Specification</th>
<th>Expression</th>
<th>UPPAAL</th>
<th>Result</th>
<th>Time (ms)</th>
<th>Mem (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Stochastic stCCSL</td>
<td>( \text{check} \triangleq \text{TurnR} &amp; \text{delay}(\text{Exp}(\text{rate})) )</td>
<td>UPPAAL</td>
<td>( \text{Pr}[\leq 100](\langle \rangle \text{a.check}) )</td>
<td>valid</td>
<td>3.91</td>
</tr>
<tr>
<td>b</td>
<td>Stochastic stCCSL</td>
<td>( \text{judge} \triangleq \text{check} &amp; \text{delay}(\text{Normal}(50, 5)) )</td>
<td>UPPAAL</td>
<td>( \text{Pr}[\leq 100](\langle \rangle \text{b.judge}) )</td>
<td>valid</td>
<td>4.01</td>
</tr>
<tr>
<td>c</td>
<td>Stochastic stCCSL</td>
<td>( \text{decide} \triangleq \text{judge} &amp; \text{delay}(\text{Normal}(50, 5)) )</td>
<td>UPPAAL</td>
<td>( \text{Pr}[\leq 40](\langle \rangle \text{c.decide}) )</td>
<td>valid</td>
<td>16.0</td>
</tr>
<tr>
<td>d</td>
<td>Stochastic stCCSL</td>
<td>( \text{exclude}(A, C) \preceq \text{join}(B, C) )</td>
<td>UPPAAL</td>
<td>( \text{Pr}[\leq 20](\langle \rangle \text{d.c2}) )</td>
<td>valid</td>
<td>3.03</td>
</tr>
<tr>
<td>e</td>
<td>Stochastic stCCSL</td>
<td>( \text{detach}(A, B) \preceq \text{join}(B, C) )</td>
<td>UPPAAL</td>
<td>( \text{Pr}[\leq 10; 1000](\langle \rangle \text{e.success}</td>
<td>t = 0) )</td>
<td>valid</td>
</tr>
<tr>
<td>f</td>
<td>Stochastic stCCSL</td>
<td>( \text{join}(A, C) \preceq \text{join}(B, C) )</td>
<td>UPPAAL</td>
<td>( \text{Pr}[\leq 10; 10000](\langle \rangle \text{f.success}</td>
<td>t &gt; 0.1) )</td>
<td>valid</td>
</tr>
<tr>
<td>g</td>
<td>Stochastic stCCSL</td>
<td>( \text{ctrlstart} \triangleq \text{ctrlsign} &amp; [0, 30] )</td>
<td>UPPAAL</td>
<td>( \text{Pr}[\leq 400](\langle \rangle \text{g.ctrlstart}) )</td>
<td>valid</td>
<td>3.94</td>
</tr>
<tr>
<td>h</td>
<td>Stochastic stCCSL</td>
<td>( \text{ok}<em>\text{turn} # \text{risk}</em>\text{turn} # \text{slowdown} )</td>
<td>UPPAAL</td>
<td>( \text{Pr}[\leq 10; 5000](\langle \rangle \text{h.ok_turn}</td>
<td>e = \text{true}) )</td>
<td>valid</td>
</tr>
</tbody>
</table>

![Fig. 8: Autonomous Vehicle behaviors model in UPPAAL-SMC](image_url)

Session DPCC: Data Privacy and Cloud Computing
Using Blockchain to Preserve Chain of Custody (CoC): Cloud Forensics Analysis

Georg Grabner¹*, Ali Ahmed², Nilufar Baghaei³
¹ School of Computing, University of Liverpool, United Kingdom
² School of Engineering and Computer Science, Victoria University of Wellington, New Zealand
³ School of Information Technology and Electrical Engineering, University of Queensland, Australia

¹ Corresponding Author, georg@grabner.nl

Abstract

In the era of cloud computing, the focus of cyber criminals has shifted from traditional IT infrastructures to the cloud. Cyber forensic investigations in the cloud present several challenges due to legal obligations, technical limitations, and the dynamic nature of cloud resources. For instance, access to digital assets is crucial for practical analysis but is often hindered, as investigators may not have complete access to disk images or log files. The mainstream forensic frameworks and solutions for identifying, retrieving, and preserving evidence in the cloud are lacking. Yet, NIST has identified 65 cloud forensics challenges, with 13 related to log files alone. The preservation and integrity of forensic evidence are the backbones of a digital forensic investigation, and the chain of custody proves the authenticity of the evidence. However, cloud environments are not designed to handle digital evidence with integrity, leading to severe issues when presenting evidence in a court of law. This paper delves into the challenges of cloud forensics investigation, focusing on log files, and aims to identify potential solutions and frameworks to support the chain of custody. For that purpose, the authors evaluate the suitability of blockchain technology for maintaining a proper chain of custody of log files in the cloud. A prototype is implemented to serve as a proof of concept.

1. Introduction

Digital forensics aims to identify, preserve, analyse, and present digital evidence in a legally acceptable manner, with the preservation and retrieval of digital evidence being integral components of the process [1]. In traditional forensic investigations, investigators are granted full access to the devices involved in a crime, such as a PC. Maintaining a proper chain of custody is critical to the process [2]. However, the complexity of preserving evidence and maintaining a chain of custody is heightened in emerging computing environments, such as the cloud and the Internet of Things, where evidence may be stored in many different places [2].

Digital forensics professionals must be familiar with cloud computing technologies and the laws and regulations surrounding data privacy, protection, and access to determine the most effective and legally compliant way to access the needed evidence [3]. Additionally, cloud forensics presents significant challenges, particularly concerning log files and maintaining the integrity and chain of custody [4, 5]. Currently, there are no prevalent solutions or frameworks for cloud forensics.

This paper addresses two research questions: RQ1) What are the challenges related to the chain of custody in cloud forensics investigations, particularly concerning log files? And RQ2) How can blockchain technology be implemented to address these challenges?

2. Background and Literature Review

Digital evidence integrity and chain of custody are vital in digital forensics [6]. However, cloud computing’s multi-tenant structure and distributed processing present challenges for collecting and preserving digital evidence, compromising the admissibility of evidence in court if the chain of custody is altered [6]. Cloud models also impact the access to resources during a forensic investigation, with SaaS models relying solely on Cloud Service Providers for forensic data [6].

Few frameworks address the challenges of cloud...
forensics analysis, particularly in maintaining the chain of custody. For instance, a Forensic Readiness Approach suggests a reference architecture consisting of a forensic database component for gathering and storing evidence and a core module that encrypts data and manages the analysis process, including the chain of custody [7]. However, no technical solutions are offered. Similarly, a client-server model captures all data and log files and stores them on a forensic server outside the IaaS cloud environment but fails to address the requirements for preserving a proper chain of custody [4]. Another methodology proposes selecting appropriate cloud services to support the forensic process, focusing on IaaS services, without offering technical solutions [8].

Blockchain is a decentralised technology that uses a distributed peer-to-peer network to store data in a secure ledger without the need for a central authority [9]. Transactions are validated and synchronised by all nodes in the network using cryptographic techniques to ensure the immutability of the data [10,11]. Blockchain applications include cryptocurrency, where it forms the backbone of a secure transaction network.

Several studies have used Blockchain technology to address the challenges of maintaining the chain of custody in forensics [12,13]. For instance, the study in [12] proposes using Blockchain to ensure the immutability of the evidence and the authenticity of the chain of custody process. In their model, evidence is admitted to the network, and all participants are uniquely identified and authorised. Similarly, the study in [13] presents a Java Blockchain implementation for preserving digital evidence in cloud environments.

Forensics in cloud environments pose several challenges, including the absence of direct access to the infrastructure and the volatility of the evidence. NIST has identified 65 issues related to digital forensics in the cloud, and log files and chain of custody are considered the main challenges [14]. Log files, one of the most crucial evidence artefacts in an investigation, must be retrieved, preserved, and maintained securely with their hash values to ensure their integrity. Research in cloud forensics has shown that log files retrieval, preservation, and maintaining the chain of custody are the primary concerns in this field [6,7,15,16].

Blockchain technology offers several benefits for maintaining and tracking the chain of custody in forensic investigations. It provides integrity, transparency, authenticity, security, and auditability, making it a reliable source of evidence in court. The consensus mechanism in blockchain ensures that multiple nodes verify every transaction, maintaining the integrity of the data. Blockchain technology is transparent, allowing easy tracking of the chain of custody. The authenticity of the data is guaranteed as every transaction is verified and authenticated by multiple nodes. The technology also provides a high level of auditability, ensuring the accuracy and integrity of the data stored on the blockchain.

In [12], a blockchain-based solution for digital forensics chain of custody is presented, with a prototype created using Hyperledger Composer and acceptable performance results. However, the work lacks a complete and optimised end-to-end solution. Similarly, the work in [17] explores the potential of blockchain technology in supporting digital forensics and investigations, discussing various blockchain implementations and introducing the concept of Digital Witnesses. While the authors conclude that blockchain-based CoC offers a new level of forensic readiness, more work is needed in data governance and standardisation. Finally, [18] presents a blockchain-based system for secure chain-of-custody transfer and record-keeping, which offers tamper-proof transaction records and increased process efficiency. However, the system does not store the actual evidence.

In [19], a digital chain of custody framework is proposed to integrate blockchain technology into digital forensics. The framework uses smart locks to store evidence, private blockchain to store evidence metadata, and a peer-to-peer network for communication. The framework is implemented using Ethereum nodes, and its performance is evaluated, showing acceptable transaction throughput. Similarly, [20] proposed a blockchain-based e-Chain-of-Custody (e-CoC) ledger, managed by a trusted entity, to ensure the integrity of digital evidence in cyber investigations. The e-CoC ledger provides a tamper-proof record of the chain of custody of digital evidence and can be easily implemented for forensic software developers. While some blocks are sent to a secure public blockchain, the work highlights the need for further research into data governance and standardisation of the admissibility of digital evidence.

In [21], an Information Chain of Custody Model based on blockchain technology is proposed to ensure privacy for sensitive health data. Limitations were identified, but the proposed solution provides traceability and control over information by the owner. The work in [22] proposes MF-Ledger, a blockchain-enabled digital multimedia forensics investigation architecture that addresses issues and challenges in digital forensic investigations. The implementation of MF-Ledger shows the potential of blockchain technology to protect and secure the digital forensic chain of custody. In [23], the authors suggest standardising smart contracts for
a secure and reliable chain of custody process, but note that a holistic approach, including robust evidence and participant management, is necessary. The work in [24] evaluates the effectiveness of fuzzy hashing algorithms in preserving the integrity of digital evidence in image forensics compared to conventional cryptographic hash algorithms, showing that fuzzy hash-based blockchain effectively supports the chain of custody process.

3. Proof of concept

A Chain of Custody process must adhere to the generally accepted four principles of digital evidence namely 1) **Principle 1**: No action taken by anyone should change the evidence, 2) **Principle 2**: In circumstances where access to the original data is needed it must be clear what the implications will be, and provide a statement of evidence why these actions have to be done. Also, the individual must be competent to handle the data, 3) **Principle 3**: An audit trail of all events or processes being executed on the evidence must be created and preserved and be able to be examined by an independent third party. And must also be able to achieve the same results with the same process; 4) **Principle 4**: The person in charge of the investigation must ensure the application of these principles.

Given our investigations, the following requirements regarding a chain of custody process are identified 1) Integrity: the evidence has not been altered or corrupted during the transfer; 2) Traceability: the evidence must be traced from the time of its collection until it is destroyed, 3) Authentication: all the entities interacting with a piece of evidence must provide an irrefutable sign as recognisable proof of their identity, 4) Verifiability: the whole process must be verifiable from every entity involved in the process, and 5) Tamper-proof: Changeovers of evidence cannot be altered or corrupted. To demonstrate those requirements and based on the principles outlined before, a prototype was developed with three main components: a private blockchain, an evidence retrieval agent, and an evidence storage and registration agent. The overall design of the prototype is shown in Figure 1.

The prototype has been implemented using Python on Ubuntu 18.04.3 TLS. For the Blockchain part, Ethereum and Hyperledger were used due to the availability of good documentation and community support. It is worth noting that the Hyperledger blockchain implementation is based on Hyperledger fabric 1.2 with Composer 2.0. The prototype uses the OwnCloud \(^1\) platform as a PaaS server from which the forensic evidence in the form of log files is retrieved.

\(^1\)https://owncloud.com

Security-wise, various serious problems were discovered. The root cause of these issues is that the authentication details were embedded in the scripts, which is highly dangerous and must not be allowed in operational settings. A part of the technical evaluation entails analysing the system’s operational efficiency. To
this end, thirty-six evidence records are sent to the Hyperledger REST interface in batches. The time taken to record these entries onto the blockchain is measured, as shown in Figure 4.

An experiment was conducted to assess the impact of record size on the system’s performance during run-time, as depicted in Figure 4.

The experiment yielded two expected outcomes. Firstly, the processing time also increased as the number of records increased. This is because blockchain operations are computationally intensive and require more time to compute hash values as the number of log files increases. Secondly, we expected that the system’s overhead would remain consistent regardless of the size of the records. The results confirmed our expectations, as the overhead introduced by the system remained consistent regardless of record size. The results in Figure 5 align with the average latency of Ethereum, and Hyperledger reported in [25]. This is a promising finding as it suggests that the system’s additional functions for evidence gathering and handling are lightweight and do not significantly impact system performance. Overall, the results of this experiment demonstrate the importance of considering record size when assessing the performance of blockchain-based systems for forensic investigations.

The purpose of the prototype was to demonstrate proof of concept, not to create a fully functional system. Hence, code review and run-time performance analysis were deemed sufficient evaluation criteria. Given the developed system, its evaluation, and the literature survey in this paper, the original research questions are answered as follows. **RQ1 - What are the current challenges regarding the chain of custody, particularly concerning log files in the cloud forensics investigation?**

The identification and preservation phase and maintaining the chain of custody of log files in cloud environments pose several challenges, per the literature survey and examination of prototype results. These challenges include: 1) Volatility of data: The data in cloud environments is highly volatile due to the elastic nature of the cloud, making it challenging to preserve evidence as virtual machines or docker images are removed or deployed; 2) Overwhelming amount of logging data: In an enterprise IaaS environment with hundreds or thousands of virtual machines, each virtual Windows machine alone can generate 86 different log files, leading to a significant challenge in monitoring and preserving all log files, 3) Multi-tenancy and Privacy concerns: In multi-tenant environments like cloud, isolating and preserving evidence without interfering with other tenants’ data or processes is challenging, along with privacy concerns, 4) Inaccessibility and Multi-Jurisdiction issues: In cloud environments, investigators do not have unrestricted or physical access to cloud storage, making it difficult to determine the location of physical data due to the distributed nature of cloud storage. Additionally, suppose the data resides in a different geographical location. In that case, jurisdiction also becomes a challenge, and 5) Integrity of log data and lack of logging standard: Maintaining the integrity of log data during preservation is challenging due to the different logging formats of cloud infrastructure and applications and the lack of standardisation on the minimum requirements for logging and retrieval
of log file metadata. To ensure integrity, SHA-256 is recommended.

The proposed solutions in [12, 26] primarily concentrate on preserving the chain of custody in the evidence handling process during the investigation phase, not during the retrieval and evidence registration phase, where evidence is submitted manually to the blockchain. On the other hand, solutions like those in [4, 27] focus on the automatic retrieval and storage of logs. However, these solutions do not specifically address the challenges in the chain of custody process.

RQ2 - How can the characteristics of blockchain technology provide solutions for these challenges, and in what way can blockchain technology be implemented to address these challenges?

The theoretical ideal solution for maintaining the Chain of Custody is a blockchain due to its distributed nature and cryptographic chaining of each transaction. This provides immutability and provenance for each evidence record and the necessary audit trails requirements such as integrity, transparency, authenticity, security, and auditability of digital evidence [12]. For a blockchain-based forensic network, a private blockchain is the best option, as it can be placed outside the client cloud environment under the control of a third-party investigator or auditor. Regarding blockchain implementations, Hyperledger Fabric and Ethereum are the best and most widely used open-source options. Different parties, such as cloud service providers, customers, and investigators, can be implemented as participating organisations within the blockchain, and strict security measures can be applied to restrict participant actions.

4. Conclusion and Future work

The current design of cloud environments does not prioritise the integrity and handling of digital evidence. However, incorporating Blockchain technology holds promise in addressing the challenges of maintaining a chain of custody for log-file evidence in the cloud. The proposed system addresses the challenge of volatile data by storing the log-file contents in a database. The system acknowledges that the large volume of data still poses a challenge but attempts to address multi-tenancy and privacy by assigning unique IDs for virtual machines and clients and storing evidence records in separate databases or servers for each customer.

The proposed solution partially addresses the challenges of inaccessibility and multi-jurisdiction by retrieving the evidence to a forensic server database accessible to the investigators. The implementation also partially solves the issues of log-data integrity and the lack of a logging standard by using a unique evidence ID and hash generated during log-file retrieval and saving metadata with the evidence record. The forensic server also records information about who, when, and where the evidence was retrieved. Despite this, the large volume of log data remains a challenge, but blockchain implementations such as Bitcoin [10, 11] have demonstrated their ability to handle vast amounts of transactions.

Future work in this research field should focus on finding solutions to the high latency experienced by the system as the number of records increases dramatically, which is crucial in digital forensics investigations where data being acquired could be in terabits.

References


DNN Inference Task Offloading Based on Distributed Soft Actor-Critic in Mobile Edge Computing

Wenxiu Xu  
College of Computer and Electronic Information  
Guangxi University  
Nanning, China  
xwx_6838@163.com

Ningjiang Chen*  
College of Computer and Electronic Information  
Guangxi University  
Nanning, China  
chnj@gxu.edu.cn

Huan Tu  
College of Computer and Electronic Information  
Guangxi University  
Nanning, China  
th_1998@163.com

Abstract—In mobile edge computing, DNN-driven intelligent inference service is highly sensitive to latency. Recently, collaborative inference between user devices and Edge Servers (ESs) based on DNN partition has been used in service acceleration. However, due to the limited computing resources of ESs, there is resource competition between concurrent requests, resulting in the partition tasks cannot be offloaded to ESs in time. Therefore, it is necessary to design an efficient offloading scheme for partition-based concurrent inference tasks. Existing task offloading schemes based on Deep Reinforcement Learning (DRL) can solve complex decision-making problems in high-dimensional state space, but there are problems such as insufficient sample diversity and easily falling into local optimum. Therefore, we propose a collaborative DNN inference task offloading scheme based on distributed Soft Actor-Critic (SAC). It supports SAC Agents to explore samples in parallel and share learning experiences, and improves the randomness of the policy through the maximum entropy mechanism to avoid falling into local optimum, thus achieving efficient offloading of concurrent partition tasks. Experimental results on DNN benchmarks show that compared with the baseline schemes, the average service latency of our scheme is reduced by more than 18.3%, and it has a higher convergence speed and task success rate, which can make ESs achieve load balancing.

Keywords-component; mobile edge computing; DNN inference; task offloading; distributed SAC; experience sharing

I. INTRODUCTION

Edge intelligent inference services driven by Deep Neural Networks (DNN) are rapidly spreading on Internet of Things (IoT) devices [1], such as image recognition, video processing, and augmented reality, which are highly sensitive to latency. The traditional method uses the powerful computing power of the cloud computing center to provide low-latency DNN inference services [2], but the long-distance transmission of media data will generate high transmission latency and energy consumption, while mobile edge computing (MEC) will computing resources sink to the edge near the data source, providing users with more agile service response by deploying Edge Servers (ESs) [3].

Recently, collaborative inference between User Devices (UDs) and ESs based on DNN partitioning in MEC has been widely used in service acceleration [4], because UDs can efficiently process the frontend part of DNN requests, greatly reducing data transmission latency. However, the computing resources of ESs are limited, and there is resource competition between partition-based concurrent inference tasks, which may lead to uneven task allocation between ESs and even ESs overload, thus failing to achieve the acceleration effect of collaborative inference. Therefore, how to offload partition-based concurrent inference tasks to ESs with limited resources to reduce service latency and achieve load balancing among ESs has become an urgent problem to be solved.

Recent task offloading methods model the offloading process as a Markov Decision Process (MDP), and use Deep Reinforcement Learning (DRL) technology to solve the MDP problem, and then offload the tasks to appropriate ESs [5], which reduces the service latency in a MEC environment with limited resources. Because DRL can effectively exert the feature extraction ability of deep learning and the learning ability of reinforcement learning and solve the complex decision-making problems in high-dimensional state space [2]. For example, Liu et al. [6] proposed a task offloading algorithm based on Dueling Deep Q-Network (DDQN), which realized the online task offloading for service acceleration under stochastic task generation and dynamic network conditions. Wu et al. [7] modeled the offloading problem as a constrained MDP and proposed an inference task offloading algorithm based on Deep Deterministic Policy Gradient (DDPG) by using the Lyapunov optimization technique, which realized the optimal allocation of computing resources. However, most of the existing DRL-based task offloading schemes use centralized Agent exploration to continuously interact with the environment [5], which has the problems of insufficient diversity of learning experience and high exploration cost. Concurrently, Agent has low exploration efficiency and sample learning rate in the exploration process, which makes the policy difficult to converge and easy to fall into local optimum.

Therefore, to solve the problem of partition task offloading in a high concurrent MEC environment, we propose a collaborative DNN inference task offloading scheme CDO-D SAC based on distributed Soft Actor-Critic (SAC), which determines the optimal offloading decision for a set of partitioning-based concurrent inference tasks. The main contributions of this paper are as follows:

- We model the offloading problem as MDP with entropy and propose CDO-D SAC to solve it. CDO-D SAC supports SAC Agents to explore in parallel to share learning experiences for policy optimization, and periodically selects the Agent with the highest average return to update the optimal policy parameters synchronously, which solves the problems of insufficient diversity of learning experience and high cost of Agents exploration in centralized training.

DOI reference number: 10.18293/SEKE2023-150

This work is supported by the National Natural Science Foundation of China (No.62162003), and the Nanning Science and Technology project (No. 20221031).
CDO-DSAC takes the maximum entropy as the goal to improve the randomness of the policy, to avoid the policy falling into the local optimum, and obtain the offloading decision with better latency. Concurrently, it encourages Agents to explore through the automatic entropy adjustment mechanism to improve their sample learning rate and convergence speed. The experimental results show that compared with the baseline schemes, CDO-DSAC has better performance in terms of acceleration performance and reliability, and has higher convergence speed and average return, which effectively reduces the exploration cost.

II. RELATED WORK

To achieve load balancing between ESs, the method based on task offloading offloading computer tasks to appropriate ES accelerates task execution while improving resource utilization. Some studies have used traditional heuristic methods based on linear/nonlinear optimization, genetic algorithm, and game theory to achieve task offloading in MEC, and achieved good service acceleration results. For example, Chen et al. [8] modeled the task offloading problem in MEC as a mixed integer nonlinear optimization problem and designed an efficient task offloading scheme SDTO. Literature [9] proposed a distributed computing offloading scheme based on a matching game mechanism, which offloads partition-based inference tasks to the edge cloud to achieve service acceleration. However, the above methods do not have sufficient autonomous decision-making capabilities and cannot achieve the expected acceleration performance in a dynamic MEC environment.

In recent years, RL and DRL technologies play a key role in solving the above problems. For example, Xu et al. [10] designed an RL-based inference task online admission algorithm Online RL, which generates an offloading strategy for randomly arrived tasks. However, RL technology cannot cope with the decision-making problem of high-dimensional state space and lacks versatility and fast adaptability. DRL has strong feature extraction ability and learning ability, which provides a solution for task offloading problems in high-dimensional state space. For example, Literature [11] implemented a real-time offloading program based on Asynchronous-Advantage-Actor-Critic (A3C) to solve the task offloading problem in MEC stochastic environment. Ren et al. Literature [12] proposed an offloading optimization algorithm based on Proximal Policy Optimization (PPO) to solve the stochastic optimization problem of when and where tasks are offloaded.

Recent studies have applied advanced DRL algorithms to solve the task offloading problem. For example, Wu et al. [7] proposed a DDPG-based task offloading strategy to optimize resource allocation in continuous state space in the MEC environment. Literature [13] modeled the offloading problem as an MDP with constrained hybrid action space and proposed a DDPG-based offloading strategy D3PG. It optimizes computational offloading in a dynamic environment by joint task partitioning and computing power allocation. DDPG is a DRL algorithm with a deterministic policy gradient, which converges fast in continuous state space, but is not suitable for a stochastic environment.

However, the existing DRL-based task offloading schemes have two defects. First, centralized agent exploration does not consider the distributed characteristics of MEC, and there are problems of insufficient diversity of learning experience and high exploration cost. Second, in the process of policy training, there are problems of poor Agent exploration efficiency and low sample learning rate, which leads to difficult policy convergence and easy to falls into local optimum. This paper focuses on solving the above problems to improve the performance of partition-based concurrent inference task offloading in policy convergence and service acceleration while ensuring the reliability of offloading schemes in extreme MEC environments and load balancing between ESs.

III. PROBLEM MODELING

At time slot $t$, we define $I_e = \{I_{e1}, I_{e2}, \ldots, I_{en}\}$ as a set of partition-based concurrent inference tasks offloaded from UDs to ESs, and $E = \{e_1, e_2, \ldots, e_k\}$ is denoted as a set of ESs. To determine the optimal latency offloading policy, we formulate the offloading problem in the MEC network modeled as an MDP with an entropy term, where the four elements are defined as follows.

1. **State**: At time slot $t$, the system state is denoted as $S^t = (I_e, E^t, N^t)$. $I_e = \{I_{e1}, I_{e2}, \ldots, I_{en}\}$ describes the state information of partition-based concurrent inference tasks; $E^t = \{e_1, e_2, \ldots, e_k\}$ describes the workload state of ESs, there is $e_j = |c_{i,j,new} - c_{i,j,\text{max}}|$, where $c_{i,j,new} = c_{i,j,\text{max}} - c_{i,j}^t$ represents the current acceptable task calculation amount of $e_j$, determined by the maximum service capacity $c_{i,j,\text{max}}$ and the task calculation loaded in the time slot $t$-1; $N^t$ describes the network state, which means $N^t = (b^t, g^t)$, $b^t$ is the network bandwidth, and $g^t$ is the channel gain.

2. **Offloading Actions**: in the policy exploration phase, each ES can be a candidate offloading action for an inference task, expressed as $a_j = \{e_1, e_2, \ldots, e_j\}$, $e_j \in \{0, 1\}$, and there is only one $e_j=1$. Therefore, at time slot $t$, the offloading actions of a set of tasks can be expressed as $A = (a_1, a_2, \ldots, a_n)$.

3. **Reward Function**: once an offloading action is generated in the current state, the Agent will obtain a system instant reward from the environment, scoring the current offloading action $A$. The goal of collaborative offloading is to minimize the service latency of inference tasks, and the offloading actions of concurrent inference tasks will affect each other. Therefore, we define the reward as the negative value of the total service latency of the system under the offloading policy.

$$R(S, A) = - \sum_i T_i$$}

Because the DNN is divisible, we allow UDs to offload part of the inference tasks to the ESs, so the service latency $T_i$ of the inference task is composed of the inference latency of the UDs side, the data transmission latency, the queuing latency, and the inference latency of the ESs side.

4. **State-Action Entropy**: considering the influence of ESs workload state on the offloading action, the state-action entropy term $H(s' | S) = - \sum p(s' | S) \log p(s' | S)$ is added to improve the randomness of the policy while encouraging Agent exploration to avoid falling into local optimum, where $p(s' | S)$ is the probability matrix of the offloading action $A'$ under $S'$. Specifically, at time slot $t$, when multiple sets of offloading...
actions are optimal, the Agent will randomly select one, which ensures that each set of valuable offloading actions will not be ignored.

We define the behavior of generating offloading actions for a set of partition-based concurrent tasks as the collaborative offloading policy \( \pi^{*}_e \). The optimal offloading policy \( \pi^{*}_e \) can be learned by maximizing the expectation of cumulative discount reward with entropy, that is to maximize the average return, denoted as:

\[
\pi^{*}_{e} = \arg \max_{\pi_{e}} E[\sum_{t=1}^{T} \gamma^{t} (R^{t} + \alpha H(\pi(A_{t}^{e} | S_{t}^{e})))],
\]

where \( \phi \) is the policy parameter, \( \gamma \in [0,1] \) is the discounted factor, and \( \alpha \) is the temperature coefficient that controls the randomness of the offloading policy.

IV. CDO-DSAC: COLLABORATIVE DNN INFERENCE TASK OFFLOADING BASED ON DISTRIBUTED SAC

A. Overview and Workflow

The overview and workflow of CDO-DSAC is shown in Figure 1, which consists of two parts. One part is distributed deployed on each ES, consisting of Communication Manager and SAC Agent. Communication Manager is responsible for communicating with UDs and ESs and collecting system state information, such as partition inference tasks status information, ESs workload state information, and network state information, and is responsible for offloading tasks to the application container instances of each ES according to the offloading decision, corresponding to steps ①, ② and ③. SAC Agent is a DRL network developed based on maximum entropy, which can approximate the optimal latency offloading policy according to the system state information, corresponding to ④. The other part is Centralized Controller deployed at the central node of the MEC network, which includes Shared-Experience Replay Memory \( D \) and Optimal Policy Updater. \( D \) is responsible for collecting the learning experiences, average return and policy parameter information explored by each Agent, corresponding to ⑤. Optimal Policy Updater is responsible for periodically selecting the SAC Agent with the largest average return as the optimal policy according to the information collected in \( D \), corresponding to ⑥.

Figure 1. The overview and workflow of CDO-DSAC

CDO-DSAC supports SAC Agents distributed exploration and shared learning experiences. For each SAC Agent, the optimal offloading policy \( \pi^{*}_{e_{i}} \), \( \forall e_{i} \in \mathcal{E} \) can be obtained by maximizing the average return, which is expressed as:

\[
\pi^{*}_{e_{i}} = \arg \max_{\pi_{e_{i}}} E[\sum_{t=1}^{T} \gamma^{t} (R^{t}_{e_{i}} + \alpha_{e_{i}} H(\pi_{e_{i}}(A_{t}^{e_{i}} | S_{t}^{e_{i}}))))] \forall e_{i} \in \mathcal{E},
\]

where \( \phi_{e_{i}} \) is the policy parameter of the SAC Agent deployed on \( e_{i} \), \( B_{e_{i}} \) stores a batch of shared learning experiences randomly selected from \( D \), which improves the diversity of learning sample and reduces the exploration cost of each SAC Agent interacting with the environment.

To speed up the CDO-DSAC training, we set the optimal policy cycle \( \omega \) to ensure that each SAC Agent can learn the optimal offloading policy. Each iteration has an optimal policy update cycle, and a SAC Agent with the largest average return is selected as the globally optimal policy, and the policy parameters are updated through (4).

\[
\pi^{*}_{e_{i}} \rightarrow \omega \{ \arg \max_{\pi_{e_{i}}} \pi^{*}_{e_{i}} \}, \exists e_{i} \in \mathcal{E}.
\]

B. Network Structure and Update Process of SAC Agents

The network structure for the SAC Agent of each ES is shown in Figure 2, where a SAC Agent is taken as an example, with \( \mathcal{V} e_{i} \subset \mathcal{E} \). SAC Agent mainly consists of Actor, Critic, and Experience-Cache. Actor is responsible for interacting with the environment and determining the offloading action for each partition task according to the system state. Critic is responsible for evaluating the offloading policy learned by the Actor. Experience-Cache consists of Replay Memory, Mini-Batch, and Parameter Synchronizer. Replay Memory is used to store the historical learning experiences \( (S^{t}, A^{t}, R^{t}, S^{t+1}) \) learned by SAC Agent. When the learning experience reaches a certain amount, it will be uploaded to the Centralized Controller, and each SAC Agent shares the collected learning experience. Mini-Batch is used to store a batch of learning experiences randomly selected from \( D \) and is used for policy optimization. Parameter Synchronizer is responsible for synchronizing the latest policy parameters updated by the Optimal Policy Updater to the Actor and Critic so that each SAC Agent can learn the optimal policy.

Figure 2. The network structure for the SAC Agent of each ES

(1) Critic. The Critic of each SAC Agent consists of two \( Q \) networks and two target \( Q \) networks, where double \( Q \) networks can overcome the overestimation problem. \( Q \) networks take the state-action pair \( (S^{t}, \pi(A^{t}_{e_{i}} | S^{t}) \) under the current offloading policy as input, and output corresponding average return to evaluate the current policy \( \pi^{*}_{e_{i}} \), i.e. \( Q \)-value. Although the complete trajectory cannot be obtained during training, a time slot difference is usually used to approximate \( Q \)-value, which can be calculated by the following:

\[
Q^{\hat{\pi}^{*}_{e_{i}}}(S^{t}, A^{t}_{e_{i}}) = R^{t}_{e_{i}} + \gamma_{e_{i}} E[Q^{\hat{\pi}^{*}_{e_{i}}}(S^{t+1}, A^{t+1}_{e_{i}})] \forall e_{i} \in \mathcal{E}.
\]

The \( Q \) network parameters \( \theta_{j}^{q}(j = 1, 2) \) are trained by minimize
the Bellm an residual, which is expressed as:

$$J_q(\theta^j) = \frac{1}{2} E_{S',A',\epsilon_1 \sim \pi_{\epsilon_1}} (Q(\pi_{\epsilon_1}(S',A'), \epsilon_1) - \hat{Q}(S',A',\epsilon_1))^2, \quad \forall \ e_1 \in \mathcal{E}, \ j = 1, 2. \ (6)$$

(2) Actor. The Actor of each SAC Agent consists of an actor network and a target actor network. We use three fully-connected layers to fit the state information, which can output unbounded offloading actions with Gaussian distribution according to the mean and standard deviation. The activation function \(\tanh\) normalizes the offloading actions, maps them to the \((-1, +1)\) interval, and the segmented activation function Relu is identified as 0 or 1 (no or yes), the specific process is shown in the Actor in Fig. 2. The parameter \(\phi_{e_1}\) can be trained by minimizing the expected KL-divergence [14], expressed as:

$$J_\alpha(\phi_{e_1}) = E_{S',B_{e_1}} [\epsilon_{e_1} \log \pi_{e_1} (A_{e_1}|S') - Q_{e_1}(S',A_{e_1})], \quad \forall \ e_1 \in \mathcal{E}. \ (7)$$

(3) Update. Critic and Actor require multistep gradient updates to converge, a stable update target is provided using the target network, and the learning stability is improved by updating the target network through an exponential smoothing:

$$\left\{ \begin{array}{l}
\tilde{Q}^j \leftarrow r(t)Q^j + \left(1 - r(t)\right)Q^j, \quad \forall \ e_1 \in \mathcal{E}, j = 1, 2, \ r \ll 1 \\
\tilde{\phi}_{e_1} \leftarrow r(t)\phi_{e_1} + \left(1 - r(t)\phi_{e_1}, \quad \forall \ e_1 \in \mathcal{E}, \ r \ll 1,
\end{array} \right. \ (8)$$

where, \(\tilde{Q}^j\) denotes the parameter of the target \(Q\), \(\tilde{\phi}_{e_1}\) is the parameter of the target actor, \(r\) is the smoothing coefficient.

(4) Automatic entropy adjustment. Finally, we added an automatic entropy adjustment mechanism to the SAC Agent network to improve the exploration efficiency of the SAC Agent during policy training. When the offloading policy explores a new space, the optimal offloading policy is still unclear, and the \(\alpha_{e_1}\) value is increased to improve the exploration ability of SAC Agent. When a state space is learning and the optimal offloading policy is determined, the value of \(\alpha_{e_1}\) should be appropriately reduced. The loss of \(\alpha_{e_1}\) is minimized by (9), where \(H_0\) is the constant of the target entropy, and the specific solution steps are given in Algorithm 1.

$$J(\alpha_{e_1}) = E_{S',B_{e_1}} [\alpha_{e_1} \log \pi_{e_1} (A_{e_1}|S') - \alpha_{e_1} H_0], \quad \forall \ e_1 \in \mathcal{E}. \ (9)$$

V. EXPERIMENTAL VERIFICATION

A. Experimental environment and parameter settings

In a simulated MEC environment, ESs supported inference task offloading requests generated by UDs in a circular area with a service diameter of 150 m. Considering the heterogeneity of computing resources of hardware devices, 5 ESs with the computing power of 30 FLOPs/Byte and 80 UDs with the computing power of 5 FLOPs/Byte were configured in this experiment. Concurrently, we designed a set of environmental variables as the initial parameters of the experiment. The serving capacity of ESs was 30, the network bandwidth was 6Mbps, the transmission power was 20 dB, and the channel gain was 140.7+36.7 log d, to control the variables as a benchmark in the experiment. To simulate the randomness of task arrival, we constrained the system task arrival rate to a lognormal distribution [8], whose mean and variance was initialized to 2.0 and 0.7, respectively.

In the MEC environment that provides intelligent services, processing image data is the most common in DNN inference. Therefore, we selected three classic and advanced CNN models as benchmarks of the experiment, namely AlexNet, VGG16, and ResNet50, and partitioned the benchmarks according to the network structure, data volume, and UDs computing power to simulate UD sending partition-based concurrent DNN inference requests to ESs. We used Pytorch to construct AlexNet, VGG16, and ResNet50, used the Berkeley Deep Drive dataset (BDD 100k) [15] for model training, and then implemented CDO-DSAC in the environment to offload target recognition tasks. The latency threshold of the task was set according to the size and type of DNN benchmarks. We deployed SAC Agents on 5 ESs for distributed learning (i.e., \(k = 5\)). Each network in Critic and Actor was composed of an input layer, an output layer, and three fully-connected layers. The number of neurons was set to 256, 512, and 256 respectively. In the experiment, \(t\) was used as the time slot to discretize the time. Table I summarizes the main hyperparameter settings in CDO-DSAC.

In order to evaluate the performance of the CDO-DSAC, we selected the following four offloading schemes as baseline comparison schemes:

(1) DDPG [7]: A DRL algorithm based on Deep Deterministic Policy Gradient, which is a commonly used task offloading method in the MEC;

(2) Online RL[10]: A RL-based task offloading algorithm to solve the problem of inference task offloading in MEC;

(3) Greedy: It selects the ESs with the smallest predicted

---

**Algorithm 1: Distributed SAC-based Partition task Offloading Algorithm**

**Input:** System state \(S\), number of episodes \(e\), number of initial exploration, Mini-Batch \(B_{e_1}\), Shared-Experience Replay Memory \(D_e\), Replay Memory \(D_e\), optimal policy update period \(e_0\).

**Output:** \(\phi, \theta_1, \theta_2\), Offloading Actions \(A\).

**Initialization:** \(\phi_{e_1} = \ldots \phi_{e_1}, \theta_{e_1} = \ldots \theta_{e_1}, j = 1, 2\).

1. **while** episode is not terminated **do**

   2. **for** \(i = 1, 2, \ldots, k\) in parallel **do**

      3. **while** initial exploration is not terminated **do**

         4. Input \(S\) into Actor and get \(A_{e_1}\);

         5. Get reward \(R_{e_i}\) and next state \(S_{e_1}\);

         6. Set \(D_{e_1} \leftarrow D_{e_1} \cup \{(S, A_{e_1}, R_{e_1}, S_{e_1})\}\);

   7. **end while**

   8. Set \(D \leftarrow D \cup D_{e_1}\);

   9. Sample \(B_{e_1} = \{(S, A_{e_1}, R_{e_1}, S_{e_1})\}\) from \(D\);

   10. **for** \(i = 1, 2, \ldots, k\) in parallel **do**

      11. Update \(\theta_{e_1}, \theta_{e_1}\) based on \(B_{e_1}\) via (6), (7);

      12. Soft update \(\tilde{\theta}_{e_1} = \frac{1}{\tau} \tilde{\theta}_{e_1} + \frac{\tau}{\tau} \theta_{e_1}\) via (8);

      13. Update \(\alpha_{e_1}\) via (9);

      14. **if** \(e_0 \mod e_0 = 0\) **then**

         15. Select optimal policy \(\pi_{e_1}^*\);

         16. Update \(\phi, \theta_1, \theta_2\) via the optimal policy \(\pi_{e_1}^*\);

         17. Update \(\phi_{e_1} = \phi, \theta_{e_1} = \theta_1, \theta_{e_1} = \theta_2\);

   18. **end if**

19. **end while**
service latency for offloading, which is the default task offloading strategy of many cluster management systems.

(4) Random: It randomly offloads inference tasks to the ESs side. It is the most primitive and easiest-to-think classic offloading algorithm, and it is also a commonly used comparison object in the field of task offloading [16].

<table>
<thead>
<tr>
<th>TABLE I. MAIN HYPERPARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>Optimal policy update period</td>
</tr>
<tr>
<td>Optimizer</td>
</tr>
<tr>
<td>Learning rate of Actor</td>
</tr>
<tr>
<td>Learning rate of Critic</td>
</tr>
<tr>
<td>Discount factor</td>
</tr>
<tr>
<td>Temperature coefficient</td>
</tr>
<tr>
<td>Learning rate of αe</td>
</tr>
<tr>
<td>Target smoothing coefficient</td>
</tr>
<tr>
<td>Total number of episodes</td>
</tr>
</tbody>
</table>

Unless otherwise specified, the hyperparameters involved in the above comparison schemes are consistent with the CDO-DSAC strategy, and each data point in the experimental results is the average of 10 repeated experiments.

B. Convergence analysis

We used 10⁵ episodes to train these 5 schemes and compared their convergence. As shown in Figure 3 (a), the solid curve and the shadow area correspond to the mean and standard deviation of the average return of the five schemes, respectively, where the return of CDO-DSAC is the mean of the average return of all SAC Agents. When the episode is 3.96×10⁴, CDO-DSAC is close to convergence. Compared with DDPG and Online RL, the convergence speed is increased by 21.1% and 37.5% respectively, and CDO-DSAC can obtain a higher average return. This is because CDO-DSAC based on distributed SAC can learn more experience in less sample space, and SAC Agent based on maximum entropy has stronger exploration ability, and its action selection is more random, to avoid falling into local optimum so that CDO-DSAC can achieve convergence faster and have higher average return. However, the average return of Greedy and Random schemes always hovers around the initial value for they have no learning ability.

Figure 3 (b) shows the exploration cost of CDO-DSAC, DDPG, and Online RL under different task arrival rates. The exploration cost is the number of episodes required to explore when the strategies converge. It can be seen that as the task arrival rate increases, the system state space and the offloading action space also increase, and the exploration cost of the three offloading schemes gradually increases, while the exploration cost of CDO-DSAC is significantly lower than that of DDPG and Online RL. This is because CDO-DSAC supports distributed learning, which can ensure that each SAC Agent can achieve the optimal average return in a cycle, and SAC Agents share the learning experience obtained through exploration, thus reducing the exploration cost of each Agent, and more sufficient experience data can also help Agents achieve convergence faster. Concurrently, the cumulative discount reward based on maximizing entropy can improve the exploration efficiency of SAC Agents, so that CDO-DSAC has a higher sample learning rate, thus accelerating its training speed and reducing the exploration cost.

C. Accelerating performance evaluation

To evaluate the acceleration performance of CDO-DSAC under different task arrival rates and ESs service capacities, we conducted experimental statistics on the average service latency of five offloading schemes on three DNN benchmarks. It can be seen from Figure 4 that compared with the four comparison schemes, CDO-DSAC has the lowest average service latency under different task arrival rates, showing better inference acceleration performance and meeting the latency requirements of benchmarks. However, when the task arrival rate exceeds 1.2, most of the baseline schemes cannot meet the latency requirements. Especially when the task arrival rate is as high as 2.0, CDO-DSAC shows a more obvious acceleration advantage, and its average service latency is reduced by more than 18.3% and 36.2% compared with DDPG and Random, respectively. Because the task arrival rate is large at this time, the computing resources of ESs are limited, and there is fierce resource competition among concurrent inference tasks, resulting in the average service latency of the baseline schemes not meeting the requirements. CDO-DSAC based on distributed SAC fully considers the impact of ESs load state and service latency on offloading actions. It can encourage SAC Agents to offload tasks to ESs with lower workloads to obtain higher returns, achieve load balancing among ESs, improve resource utilization, and reduce queuing latency.

Figure 4. Comparison of service latency under different task arrival rates.

It can be seen from Figure 5 that the average service latency of CDO-DSAC under different ESs service capacities is always the lowest, and is less affected by the change of service capacity, showing better acceleration effect and stability than the comparison schemes. When the service capacity is 40, the average service latency of CDO-DSAC is 19.9% and 38.5% lower than that of DDPG and Random, respectively, because the SAC Agents based on maximum entropy can improve the randomness of the strategy, so that CDO-DSAC can train a higher return and better offloading strategy. DDPG based on deterministic strategy is easy to fall into local optimum. On the other hand, when the service capacity is less than 35, the average service latency of the four comparison schemes is greatly affected by the service capacity, and most of them do not meet the latency requirements of benchmarks, because the service capacity of ESs is extremely limited, resulting in serious resource contention between concurrent inference tasks. CDO-
DSAC can effectively alleviate the resource competition between tasks to reduce queuing latency.

**Figure 5.** Comparison of service latency under different service capacities

**D. Reliability evaluation**

We conducted experimental statistics on the task success rate of three DNN benchmarks under different task arrival rates and serving capacities. It can be seen from Figure 6 that the task success rate of CDO-DSAC under different task arrival rates is higher than that of the other four comparison schemes, showing higher task offloading reliability. Especially when the task arrival rate is as high as 2.0, CDO-DSAC shows more obvious advantages, and its task success rate is more than 18.9% and 22.4% higher than DDPG and Online RL, respectively. Because the CDO-DSAC strategy based on distributed SAC can fully consider the impact of ESs state and serving latency on offloading actions, on the one hand, it balances the workload between ESs and improves resource utilization; on the other hand, it effectively alleviates the resource competition in the concurrent environment and reduces the queuing delay, so that CDO-DSAC can improve the task success rate of concurrent inference tasks with limited ESs resources.

**Figure 6.** Comparison of task success rate under different task arrival rates

It can be seen from Figure 7, the task success rate of CDO-DSAC strategy is higher than 85% under different ESs serving capacities. Compared with the other four baseline comparison methods, CDO-DSAC shows higher task offloading reliability. Especially when the serving capacity is 25, the task success rate of CDO-DSAC is 19.8% and 33.4% higher than that of DDPG and Random respectively, because the serving capacity of ESs is extremely limited at this time, it is easy to cause task failure due to service overload. CDO-DSAC can balance the load between ESs to reduce resource contention caused by resource constraints and meet the latency requirements. Therefore, it has high task success rate and reliability in extreme environments.

**Figure 7.** Comparison of task success rate under different service capacities

**VI. CONCLUSION**

In this paper, we model the DNN inference task offloading problem as an MDP with entropy and propose the offloading schedule CDO-DSAC based on distributed SAC to solve the MDP problem. CDO-DSAC is a distributed offloading scheme based on the maximum entropy mechanism. It encourages Agents to optimize in more samples by improving the randomness of exploration, avoiding the policy falling into local optimum. The learning experience can be shared among Agents to better optimize the network, thereby expanding the scale of learning experience data, and reducing the cost of exploration. The experimental results show that CDO-DSAC is superior to the baseline comparison schemes in convergence performance, acceleration performance, stability, and reliability, and has good inference acceleration and load balancing effects. In future work, we will further study how to reduce the total energy consumption of devices in the MEC environment while improving acceleration performance.

**REFERENCES**


Bindox: An Efficient and Secure Cross-System IPC Mechanism for Multi-Platform Containers

Yuxin Xiang¹, Bing Deng², Hongyu Zhang², Randy Xu², Marc Mao², Yun Wang¹, Zhengwei Qi¹
{xiangyuxin,yunwang94,qizhwei}@sjtu.edu.cn, {bing.deng,hongyu.zhang,randy.xu,marc.mao}@intel.com
¹ Shanghai Jiao Tong University, ² Intel Corporation
Shanghai, China

Abstract

Containerization is widely used for isolation in various applications because it is lightweight, scalable, and portable. In modern distributed systems, seamless inter-process communication (IPC) between multi-platform containers is essential for a range of applications and services, including microservices, cloud computing, and Internet of Things (IoT) devices. However, secure and efficient communication between containers on the same host is challenging, especially when different operating systems are involved.

This paper introduces Bindox, a lightweight, efficient, and secure IPC mechanism that enables seamless communication across multiple platforms, including Android and Linux. Bindox uses shared memory for data transfer and implements a stable client-server architecture, ensuring high performance and ease of maintenance. Additionally, Bindox provides a robust security mechanism that guarantees confidentiality, integrity, and availability of the communication channel. Experimental results demonstrate that Bindox outperforms existing networking and IPC methods in terms of memory use, latency, and CPU usage, making it a promising solution for efficient and secure communication between multi-platform containers.

1 Introduction

Containerization has emerged as a transformative technology in modern computing due to its ability to provide an isolated runtime environment that is lightweight, scalable, and highly portable. In modern distributed systems, IPC between containers of different OS images has become increasingly important. Seamless communication between these multi-platform containers can enable a wide range of applications and services, such as microservices, cloud computing, and Internet of Things (IoT) devices [7].

However, fast and secure communication between containers on the same host can be challenging, particularly when different operating systems are involved. Traditional container communication methods on the same host can be classified into two categories: (a) networking, which includes using Docker networking and sharing the host network namespace; (b) IPC, which includes files, shared memory, UNIX sockets, pipes, semaphores, shared memory, and message passing. All these methods have their respective pros and cons (Section 2) and fail to strike a balance between performance and security. Inappropriate settings of both methods can introduce significant hazards to the system, which can be fatal when doing cross-OS interactions.

In recent years, several studies have attempted to address these challenges by introducing new IPC mechanisms [4, 6, 8, 16]. However, these mechanisms are often limited in terms of their compatibility with container environments or their ability to provide both high performance and security [15].

To address these challenges, in this paper, we introduce Bindox, a fast, secure, lightweight, and cross-system IPC mechanism that enables communication on and between multiple operating systems, including Android and Linux. Bindox leverages shared memory for data transfer to achieve high performance with zero-copy, which significantly reduces the data transfer overhead and improves performance. Bindox also provides a robust security mechanism that ensures the confidentiality, integrity, and availability of the communication channel.

To demonstrate the effectiveness of Bindox, we compare it to existing container-on-the-same-host communication methods, and the results show that Bindox shortens the transmission latency by 40% on average.

Bindox offers a promising solution for efficient, secure, and cross-platform communication between heterogeneous containers on the same host, which is critical for application performance, security, and cluster scalability. Compared to other solutions, Bindox has several advantages:

- Lightweight and flexible: Bindox is compatible with container environments and is suitable for use in various contexts and applications.
- High efficiency: Bindox achieves zero copy for data transfer, which reduces data transfer overhead and improves performance.
- Stable architecture: With a clear client-server architecture, Bindox simplifies the process of developing and maintaining the system. Unlike shared memory, users do not need to consider complex concurrency synchronization issues.
- Robust security: Bindox provides a robust security mechanism that ensures the confidentiality and integrity of communication between processes. This mechanism helps to prevent unauthorized access, data tampering, and other security threats.
- Cross-OS compatibility: Bindox supports cross-OS communication, including Android and Linux.
2 Background and Related Work

2.1 Cross-OS Communication

Communication between different operating systems is a ubiquitous requirement for modern applications. In fact, many applications critically depend on services hosted on the other OS to function properly. The seamless communication between such diverse services is essential for the development of robust and feature-rich applications.

The cloud platform is a typical example of cross-OS communication. With the emergence of various cloud-native technologies, such as Docker and Kubernetes, cloud computing has been widely adopted [12, 17], and the performance of IPC is crucial for achieving efficient cloud services. To extend the computing power of mobile devices and enable the cloud-based execution of Android applications, various approaches have been proposed for Android cloudification. For example, Android Emulator [2] and Cuttlefish [5] use virtual machines as isolation units to ensure that Android can run on any platform. To make the system lighter, Anbox [1] places Android into a container. Another solution, CARE [13], further cloudifies the Android system into a cloud-native system by streamlining Android services. In Android cloudification, Android applications are hosted on different operating systems, usually Linux. These VMs and containers rely on communication with the host to enable file sharing, network stack, and hardware device sharing. Moreover, the cloud Android applications can also utilize services on the host machine to function properly.

Autonomous car software serves as another example of Android-Linux communication. Typically, automotive software is divided into two main regions: automotive machine vision (AMV) and in-vehicle infotainment (IVI) [14]. The AMV region is responsible for the advanced driver assistance systems (ADAS) using a real-time Linux system [9], whereas the IVI region provides driving information and entertainment by utilizing an Android system [10] that is specifically tailored for consumer applications. Despite their distinct purpose, the two regions interact in various scenarios, sharing files, images, videos and sensor data on the same host. A common scenario is that an Android camera service transfers captured camera data to a Linux-based image analysis and processing module for vehicle and driver state analysis. Another example is that a real-time application running on Linux makes a decision; it needs to communicate to the media service running on Android, which can then notify the user through graphics or audio.

2.2 Container Communication On the Same Host

There are several methods for enabling communication between containers running on the same host, including networking and IPC methods. Networking methods encompass the use of Docker networking and sharing the host network namespace, while IPC methods include pipe, message passing, shared memory, UNIX sockets, semaphores, and signal. While these methods can provide fast and efficient communication, they fail to balance performance and security.

Networking methods allow containers to communicate directly with each other using IP addresses. However, it may cause considerable overhead due to the network protocol and stack, and it may pose security risks if not properly isolated from other networks.

While IPC methods, especially those that utilize shared memory, offer rapid and effective communication, their configuration and management can be intricate due to distinct namespaces. In addition, each IPC technique has restrictions, and inappropriate settings could result in substantial safety hazards, especially in cross-OS interactions. Thus, it is imperative to carefully consider the potential risks and carefully tailor the IPC method to the specific application and environment to ensure efficient and secure communication.

- Pipe: Pipes are half-duplex and limited to one-way communication between processes. To achieve bi-directional communication, two pipes are needed. Moreover, pipes are commonly used between parent and child processes and have a relatively small buffer size.
- Message queue: Message queues buffer size limits the amount of data to be transmitted. Moreover, message queues can lead to synchronization issues including deadlocks and priority inversion, which can negatively impact system performance.
- Shared memory: Shared memory is a high-performance IPC method that directly attaches shared buffers to a process’s virtual address space. However, the synchronization between processes is left to the responsibility of the processes themselves. Furthermore, it may leak confidential data without proper access restriction.
- UNIX domain socket: Unlike network sockets, UNIX domain sockets are based on file system path names and do not require processing through the network protocol stack, thus offering higher performance and lower latency. However, UNIX domain sockets can introduce security risks if the socket is not properly secured. It is important to ensure that the socket is properly permissioned and that appropriate access controls are in place to restrict access. Moreover, they do not support more advanced features such as multicasting, which are often used in distributed systems.
- Semaphore: Semaphores are mainly used as lock mechanisms to prevent multiple processes from accessing shared resources simultaneously. Therefore, they are mainly used for inter-process and inter-thread synchronization.
- Signal: Signals are not suitable for information exchange but are instead useful for process interrupt control, such as handling illegal memory access or killing a process.
3 Bindox: Design and Implementation

3.1 Bindox Architecture

This section details the components of the Bindox design. Aiming at a user-friendly, secure-by-design, and cross-OS adaptive architecture, Bindox takes advantage of a stable client-server model to enable efficient and scalable IPC in a heterogeneous multi-OS container environment. As shown in Fig. 1, communications in Bindox involve four components: client, server, Coordinator, and Bindox Driver. In Bindox, clients and servers can be deployed on any host process or container. Coordinator is a daemon process running on the host, managing and serving all the clients and servers. Bindox Driver is implemented as a kernel module responsible for handling the low-level communication between different processes.

Client and server: Clients and servers are responsible for initiating and responding to communication requests, respectively. One server can handle clients in different platform containers, as shown in Fig. 2. Clients and servers can be implemented with the Bindox library, which is compatible with various operating systems, including Android and Linux. Upon each server’s start, Driver creates a corresponding service node in the kernel space, which is the communication endpoint used to send messages between clients and services.

Coordinator: The Coordinator is a daemon process that serves as a central and secure registry for managing and accessing Bindox services. It maintains a table of available Bindox service references and provides service registration and discovery functionalities. The Coordinator also leverages the Bindox Driver to enforce security checks, ensuring that only authorized processes can access registered services.

3.2 Bindox Communication Model

This section will explain the communication process in Bindox in detail. Given the scenario that on an autonomous vehicle, the advanced driver assistance system (ADAS) needs to access camera data, analyze, and process the images to determine whether the driver is experiencing driver fatigue. The host system is a real-time operating system, Automotive Grade Linux (AGL) [3], which is a collaborative open-source project used by multiple car manufacturers. The image processing program runs in a Linux container, while the camera image capture program runs in an Android container. As shown in Fig. 1, the image processing program acts as a client and requests image capture data from the camera service.

Coordinator Setup: Before using Bindox for communication, the Coordinator process must be started on the host. This process registers itself as Coordinator at the Driver and serves as a service registry and search center for further Bindox communications.

Service Registration: Upon starting the camera service, the server sends the service name and registration request to Coordinator through Driver. Driver creates the corresponding service node in the kernel and its reference, then forwards the service name and service reference to Coordinator. After receiving the data, Coordinator appends the name and service reference to the local table.

Service Discovery: The image processing application uses the service name to query access to the camera service.
from Coordinator via Driver. Driver first validates the request source, user capability, and requested data integrity, then forwards the request to the Coordinator. In service discovery, Coordinator’s role is similar to Domain Name System (DNS), as it converts the requesting service name into service reference for clients. After Coordinator returns the service reference to the client, the camera service node now has two references: one in the Coordinator and one in the client. If more clients request the service in the future, the number of references to the camera service will increase accordingly.

**Service Request and Respond:** The image processing application can call the camera service through the service reference returned by Coordinator. Bindox Driver accomplishes all the low-level communication and security checks. Bindox security assurance includes user authentication, permission validation, and data integrity checking. However, all these details are hidden by Bindox Driver and Bindox user library, making Bindox concise and user-friendly.

### 3.3 Bindox Data Transfer

In addition to supporting cross-platform container communication, the most remarkable feature of Bindox is its high performance. Bindox achieves zero-copy during the transmission process, thereby minimizing the overhead associated with multiple copies.

In networking and conventional IPC, including pipes, message queues, and Unix sockets, at least two copies are needed during the message transfer. As depicted in Fig. 3a, these communication methods suffer from expensive copy operations, as at least two copies are required during message passing, which is particularly costly in frequent or high-volume communication scenarios.

To address these problems, Bindox proposes a new transfer method, **Mapping Delivery.** As shown in Fig. 3b, **Mapping Delivery** uses memory mapping to map the sender and the receiver user space buffers to the same physical memory, ensuring that data transfer from sender to receiver is only a logical copy without actual overhead. It is achieved through Bindox Driver. During communication, Bindox Driver provides a shared memory region that can be accessed by multiple processes, and establishes a mapping between memory, server’s and client’s user space addresses.

Through **Mapping Delivery,** Bindox can efficiently share memory between different processes in a secure and controlled manner. **Mapping Delivery** provides a shared memory region that can be accessed by multiple processes, and Bindox facilitates the communication and coordination between these processes. This combination allows for the creation of efficient IPC channels while maintaining a high level of security and access control. Specifically, **Mapping Delivery** provides a mechanism for allocating and sharing memory, while Bindox security design ensures that only authorized processes can access this memory and enforces strict security policies to prevent unauthorized access or tampering. This innovative approach enables the development of complex and highly performant systems while ensuring the security and integrity of shared data.

### 3.4 Bindox Security

The security mechanism of Bindox provides a robust defense against unauthorized access and helps to maintain the confidentiality, integrity, and availability of the system.

One of the innovative features of the Bindox security mechanism is the centralized Coordinator, which acts as a registry for all the available services in the system. Through Coordinator, Driver can checks the permission of clients asking for services. This enables the system to enforce fine-grained access control over the services and prevent unauthorized access or tampering.

Another innovative feature of the Bindox security mechanism is its use of permissions to control the interactions between components. It is based on the principle of least privilege, where every component of the system has only the minimum required permissions to perform its function. This is achieved through a series of access controls that limit the interactions between components. In Bindox, Bindox Driver mediates the interactions between processes and ensures that only authorized processes can access a particular service or component, and it prevents malicious components from causing harm to the system.

### 4 Evaluation

In this part, we present our evaluation of Bindox. We compare Bindox with networking and IPC methods, and try to answer the following questions:

- How does Bindox improve communication latency?
- What is the memory usage of Bindox during communications?
- What is the CPU usage of Bindox during communications?

#### 4.1 Environment Setup

**Configuration:** All the experiments are conducted on a server running Ubuntu 22.04 LTS with Linux kernel 5.15, six
Intel(R) Core(TM) i5-9400 CPUs at 2.90 GHz, and 15 GiB of physical memory.

Benchmarks: In the benchmark, the client process is held in a Docker Android container, and the server is running on the Linux host. Client and server alternately update the content of a fixed amount of memory and transfer the memory content to each other. Each iteration involves both client and server setting and passing memory once. To simulate real-world applications that often involve large-scale, precise, and secure data transfer, the memory size are set to 128MB, 256MB, 512MB, and 1GB. To ensure statistical accuracy, the iteration is set to 100 times during testing. This benchmark is designed to evaluate the efficiency and effectiveness of Bindox in handling data transfers.

Baselines: To illustrate the performance improvements brought by Bindox, we take networking and IPC methods as the baselines. In the networking baselines, the client container uses the host network. Three different network protocols are used: gRPC, TCP, and UDP*. gRPC is a modern open source Remote Procedure Call (RPC) framework that can run in any environment. It implements the same client-server architecture and security checks for communication as Bindox, but uses HTTP/2 as the underlying transport protocol. UDP* is a custom protocol built on top of UDP that guarantees transmission order and reliability, as we have found that in real-world scenarios, using UDP to transmit large amounts of data, even on the same host, often results in disorder and dropped packets. For IPC methods, we use UNIX domain socket as a baseline, as UNIX domain socket is widely used in container communication.

4.2 Transmission Latency

In this experiment, we present the average latency of message transfer iterations of Bindox.

Fig. 4 presents the time cost of different methods. We can witness that Bindox transfer latency is significantly lower than that of other communication methods. When transmitting data ranging from 128MB to 1GB, the average transfer iteration time cost of Bindox is 40.7%, 39.2%, 39.5%, and 20.8% lower than that of TCP, the suboptimal communication method, respectively. Even with extra security checks, Bindox has a significant performance advantage in handling reliable data transfer.

As Fig. 4 shows, gRPC has a much higher transmission time. This is because gRPC uses Protocol Buffers [11] as both its Interface Definition Language (IDL) and underlying message interchange format. Protocol Buffers types have language-specific implementations and require more data serialization and deserialization operations during transmission. Additionally, to ensure security, gRPC implements encryption transmission and authorization checks during function invocation, leading to a increased transmission time.

UDP* is based on the UDP protocol. It uses slicing and blocking to achieve reliable transmission, leading to a higher cost. The transmission cost of a UNIX domain socket is slightly lower than that of TCP. Although UNIX domain socket does not pass through the network stack, its bandwidth is lower than TCP. While UNIX domain socket has a significant performance advantage when transmitting small amounts of data (e.g., 1KB) compared to TCP, when transmitting large amounts of data, the time consumed by UNIX domain socket is comparable to that of TCP.

4.3 Memory usage

In this experiment, we measured the total memory usage of both client and server during transmission for each method.

As Fig. 5 presents, Bindox had a consistent memory usage pattern during message transmission. The memory consumption was stable and remained at about the size of the transmitted data. This is because Bindox uses a Mapping Delivery that avoids the need to copy data between processes.

In contrast, TCP, UDP, and UNIX domain socket all require data copying between the user space and kernel space, resulting in memory consumption that is twice the size of the transmitted data. As for gRPC, its memory consumption is observed to be highly variable during message transmission, and the average value is several times the size of the transmitted data, as gRPC requires serialization and deserialization operations during data transmission, which increases memory usage.
4.4 CPU usage

This experiment shows the total CPU usage of both client and server during transmission for each communication method.

As Fig. 6 shows, the total CPU usage of client and server of Bindox is relatively stable and lower than that of other methods. This can be attributed to the efficient design of Bindox Mapping Delivery, which utilizes shared memory to reduce the overhead of context switching and data serialization.

gRPC’s is based on HTTP/2 protocol, leading to a higher CPU cost. However, its underlying message interchange format, Protocol Buffers, alleviates the CPU usage for message serialization and deserialization. For TCP and UDP*, these protocols rely on the operating system’s network stack to process packets, resulting in frequent context switching and increased CPU overhead. Moreover, TCP and UDP* require additional overhead to handle packet fragmentation, retransmissions, and flow control. Additionally, TCP and UDP* perform checksum calculations on each packet, which also increases CPU usage. For Unix domain socket, it involves copying data between user and kernel space, and in the case of large data volumes, the CPU is polling for the kernel buffer to become available, resulting in significant CPU overhead.

5 Conclusion

Bindox offers a promising solution for efficient, secure, and cross-platform communication between heterogeneous containers on the same host. Compared to other solutions, Bindox is lightweight, flexible, and container-compatible, with a stable client-server architecture that simplifies development and maintenance. Additionally, Bindox provides a robust security mechanism that ensures the confidentiality and integrity of communication.

Acknowledgments

We thank for the reviewers’ insightful feedback. This work was supported in part by National NSF of China (No. 62141218), Shanghai Key Laboratory of Scalable Computing and Systems, and Intel Corporation.

References

Smart Contract Vulnerability Detection Based on Clustering Opcode Instructions

Xiguo Gu, Huiwen Yang, Shifan Liu, Zhanqi Cui*
Computer School, Beijing Information Science and Technology University, Beijing, China
Email:{xiguo_gu, yhw_yagol, pawn2017, czq}@bistu.edu.cn

Abstract

Smart contracts are programs running on the blockchain. In recent years, due to the continuous occurrence of smart contract security accidents, how to effectively detect vulnerabilities in smart contracts has received extensive attention. Machine learning-based vulnerability detection techniques have the advantage of not requiring expert rules. However, existing approaches have limitations in identifying vulnerabilities caused by version updates of smart contract compilers. In this paper, we propose OC-Detector, a smart contract vulnerabilities detection approach based on opcode instruction clustering. OC-Detector learns the characteristics of opcode instructions to cluster them and replaces opcode instructions belonging to the same cluster with the cluster number. After that, the similarity is calculated against the contract in the vulnerability database to identify vulnerabilities. Experimental results demonstrate that OC-Detector improves the $F_1$ value of detecting vulnerabilities from 0.04 to 0.40 compared to DC-Hunter, Securify, SmartCheck, and Osiris. Additionally, compared to DC-Hunter, $F_1$ value is improved by 0.27 when detecting vulnerabilities in smart contracts compiled by different version compilers.

Index Terms—Ethereum, Smart Contracts, Slicing, Word embedding, Clustering.

I. INTRODUCTION

The blockchain is a distributed database with characteristics of decentralization, immutability, traceability, and joint maintenance by multiple parties[1]. Smart contracts are programs that run on the blockchain, which help developers apply blockchain techniques to several fields, such as finance, education, and the internet of things. Ethereum is the most popular blockchain platform, which uses the Ethereum Virtual Machine (EVM) to execute smart contracts.

While smart contracts have given rise to a variety of applications, its stored digital assets make it vulnerable to numerous attacks. For instance, in 2016, a security vulnerability in the DAO contracts led to the loss of more than 3.6 million Ethereum tokens, with a value of about $60 million. In this case, the attacker repeatedly reentered the transfer function to steal Ethereum tokens through a reentrance vulnerability in the DAO contract[2].

Therefore, it is necessary to study effective vulnerability detection techniques to mitigate losses caused by vulnerabilities in smart contracts. Due to the rapid development of machine learning technology, machine learning-based vulnerability detection techniques for smart contracts have gained considerable attention in recent years. Specially, it is often challenging to obtain the source code of smart contracts, while contract bytecode can be directly acquired from Ethereum platforms. Hence, bytecode-based techniques for detecting vulnerabilities in smart contracts are more practical. Han et al.[3] leverage program slicing techniques, track the data flow and extract slices from contract bytecode. They also used graph embedding techniques to capture more structural information to improve the performance of detecting smart contract vulnerabilities. However, the Solidity language, which is used to write smart contract code in Ethereum, is still in the evolutionary stage. For example, 11 versions were released in 2021. Depending on the version of Solidity compilers, the same statement in the smart contract may have different opcode instructions. Therefore, it is difficult to detect vulnerabilities by using machine learning-based approaches when the model is trained on other versions of smart contracts.

To solve the above problems, we present OC-Detector, a smart contract vulnerability detection approach based on clustering opcode instructions. First, OC-Detector transforms the bytecode of smart contracts into opcode instructions, which are then embedded into vectors via word embedding techniques. Second, OC-Detector utilizes a clustering algorithm to classify vectors into several clusters based on their semantic information. After that, we construct a dataset containing 4 types of vulnerabilities by selecting representative contracts, including reentrance, timestamp-dependencies, access control, and unchecked call return values. Finally, OC-Detector slices the target contracts and represents the opcode instructions belonging to the same cluster uniformly based on the results of clustering and detects vulnerabilities by comparing the similarity between the target contract and the dataset. Experimental results demonstrate that OC-Detector is more effective than other existing approaches, including DC-Hunter[3], Security[4], SmartCheck[5] and Osiris[6]. The $F_1$ value is improved by 0.04 to 0.40. Additionally, compared with DC-Hunter, the $F_1$ value is improved by 0.27 when detecting vulnerabilities...
in smart contracts compiled by different version compilers.

To summarize, our main contributions are as follows:

- We propose OC-Detector, a smart contract vulnerability detection approach based on clustering opcode instructions. OC-Detector can alleviate false negatives and false positives caused by inconsistent compiled opcode instructions by different version compilers.

- Experiments are conducted to compare OC-Detector with DC-Hunter, Securify, SmartCheck, and Osiris. Experimental results show that OC-Detector outperforms existing approaches on a large-scale dataset.

The remainder of the paper is structured as follows: Section II provides an example to illustrate the motivation of this research. Section III presents details of the approach. Section IV describes the experiments and evaluations to validate the effectiveness of the OC-Detector. Section V analyzes threats of validity. Section VI reviews related work. Section VII concludes the paper and discusses future research directions.

II. THE MOTIVATION EXAMPLE

In practical applications, opcode instructions play a vital role in the execution of programs, and their variations can result in significant differences in program behaviors. Due to the frequent iteration of compiler versions, opcode instructions generated by different versions of compilers can vary significantly. Using existing vulnerability detection techniques to detect smart contract vulnerabilities can lead to false positives and false negatives due to inconsistencies of opcode instructions. Figure 1 shows a vulnerable contract that is compiled into various opcode instructions by different versions of compilers.

Figure 1 shows “PERSONAL_BANK[1]”, an example of a smart contract with a vulnerability, in which the vulnerability is located at line 10. The contract may execute a recursive call at line 10, which triggers a vulnerability that could cause damage to the contract caller. In line 5 of the contract, different version compilers will compile the contract to different opcode instructions. For instance, the 0.4.11 version Solidity compiler generates opcode instructions as “ISZERO, ISZERO, PUSH, JUMPI, INVALID, JUMPDEST, PUSH”, whereas the 0.4.26 version generates “ISZERO, DUP1, ISZERO, PUSH, JUMPI, PUSH, DUP1, INVALID”. When using our replicated DC-Hunter to detect opcode instructions compiled with version 0.4.26 compiler using opcode instructions generated by version 0.4.11 compiler, it fails to detect the vulnerability on line 10 of the smart contract. While by clustering the opcode instructions using OC-Detector, the opcode instructions with similar features in different versions can be clustered into the same cluster, and to make a uniform representation. This effectively eliminates the differences of opcode instruction sequences generated by the two versions of compiler. As a result, the vulnerability in the contract can be successfully detected without the influence of opcode instructions generated by different version compilers.

![Fig. 1: A motivation example of smart contract.](image)

III. SMART CONTRACT VULNERABILITY DETECTION BASED ON CLUSTERING_OPCODE INSTRUCTIONS

Figure 2 presents the proposed approach for detecting smart contract vulnerabilities based on clustering opcode instructions. First, OC-Detector converts the bytecode of a smart contract into opcode instructions, then embeds them into vectors by using word embedding models. Second, OC-Detector utilizes clustering algorithms to classify vectors of opcode instructions into several clusters based on their semantic information. After that, representative contracts are selected to construct a vulnerability dataset. The opcode instructions that belong to the same cluster in the vulnerability dataset are uniformly represented. Finally, the target contracts are sliced and the opcode instructions belonging to the same cluster are uniformly represented. The vulnerability is then detected by comparing the similarity of the target contract with those in the vulnerability dataset.

A. Opcode instruction vectorization.

Due to the difficulty of obtaining the source code of smart contracts, it is challenging to learn contract features directly from the source code. Therefore, we utilize smart contract opcode instructions to learn contract features and translate the opcode instructions into vectors.

When learning contract features, we compile the bytecode of the smart contracts and transforming them into opcode instructions. The opcode instructions are then normalized by removing the operands and applying a delimiter to separate different instructions. Next, a word embedding model is trained with the normalized opcode instruction sequences, to learn the features of opcode instructions. This step enables the extraction of meaningful features that can be used to identify similarities or differences between smart contracts and provide insights into their functionalities and potential vulnerabilities.

B. Opcode instructions clustering.

There are a total of 143 opcode instructions available in smart contracts[2]. While some opcode instructions have similar functions, they may differ slightly in their symbolic representation. For instance, PUSH1, PUSH2 and PUSH3 are all instructions used for pushing values onto the stack, but they differ in the number of bytes they push. Furthermore, it

[1]PERSONAL_BANK: https://github.com/chen2233/smartbugs/blob/master/dataset/reentrancy/0x01f8c4e3fa3edeb29e514cba738d87ce8c091d3f.sol

is imperative to note that due to different version of Solidity compilers, the opcode instructions generated for the same contract may vary. As shown in the motivation example.

To solve the above problem, OC-Detector clusters opcodes with similar functionality into the same cluster. In the clustering process, \( k \) opcode instructions are randomly selected as initial cluster centers. The distance between each opcode instruction and each cluster center is calculated, and the opcode instructions are assigned to the cluster center closest to it. For each opcode instruction assigned, the cluster centers are recalculated based on existing opcode instructions in the clusters. This process is repeated until all opcode instructions are assigned to the corresponding clusters.

### C. Opcode sequences preprocessing.

Using the opcode instruction sequences generated by contract compilation for similarity calculation can be interfered by a mass of irrelevant instructions, resulting in performance degradation of detecting vulnerabilities.

To solve this problem, OC-Detector slices the opcode sequence to reduce the interference of noise based on analyzing the dependency between opcode instructions and external data. During the opcode instructions slicing process, smart contracts containing representative vulnerabilities are first manually selected and transformed into opcode instruction sequences through preprocessing. The data dependency relationship between opcode instructions and external data in the opcode instruction sequences is then analyzed. Instructions that introduce external data and instructions that use external data are sliced according to the data dependency relationship. Finally, the sliced opcode instructions belonging to the same cluster are replaced with the cluster number, and the opcode instruction sequence is added to the vulnerability dataset.

### D. Vulnerability detection.

To effectively detect vulnerabilities in smart contracts, OC-Detector utilizes similarity calculation to detect vulnerabilities in smart contracts. To calculate the similarity, Levenstein\(^3\) distance is used to calculate the similarity between the opcode instruction sequences of two smart contract. Specifically, OC-Detector calculates the similarity between the target contract and each contract in the vulnerability dataset one by one. If the similarity exceeds a threshold \( p \), OC-Detector considers the contract to be vulnerable and output the type of vulnerability.

### IV. Experiments and Evaluations

All experiments in this paper are run on a computer with i7-6700H CPU and 16GB of memory, and the development and running environment is Ubuntu 18.04 and Python 3.9.

The following three research questions are designed to verify the effectiveness of OC-Detector.

- **RQ1:** How does the performance of OC-Detector compare to other existing tools?

  To evaluate the performance, OC-Detector is compared with DC-Hunter, SmartCheck\(^4\), Securify\(^5\) and Osiris\(^6\). SmartCheck, Securify and Osiris are open source tools, while DC-Hunter is not open-source, but it shares similarities with OC-Detector, so we implemented it according to the descriptions of the paper. In addition, to verify the effectiveness of OC-Detector in alleviating false positives and false negatives caused by inconsistent opcode instructions due to different versions of Solidity compilers, we compare OC-Detector with DC-Hunter which is similar to our approach.

- **RQ2:** How does the Word2Vec model affect the performance of OC-Detector?

  To verify the impact of the word embedding model trained by opcode instruction sequence for vectorizing opcode instructions, we replace the Word2Vec model trained by opcode instruction sequence with the default Word2Vec model (denoted as OC-Detector\(^*\)) and compare it with OC-Detector.

---

\(^3\) Levenstein: https://github.com/ztane/python-Levenshtein
\(^4\) SmartCheck: https://github.com/smartdec/smartcheck
\(^5\) Securify: https://github.com/eth-sri/securify
\(^6\) Osiris: https://github.com/christoftorres/Osiris
• RQ3: How does the clustering opcode instructions affect the performance of OC-Detector?

To verify the impact of the clustering algorithm used in OC-Detector, we directly calculate the similarity between the target contract and the contract in vulnerability dataset without clustering (denoted as OC-Detector w/o clustering) and compare it with OC-Detector.

A. Experimental dataset.

The experimental objects utilized two widely used datasets, include Xblock[7] and Smartbugs[8], which have been used in prior smart contract vulnerability detection studies[7][8]. The Xblock dataset comprises a total of 149,363 smart contracts, from which 10,000 contracts were randomly selected for training the Word2Vec model. Another 5,000 contracts were randomly chosen as the target contracts to evaluate the effectiveness of OC-Detector. Since the Xblock dataset lacks vulnerability information, we utilized Slither[9] to analyze the 5,000 target contracts and annotate their vulnerability labels. Empirical studies have shown that Slither is the most effective static analysis tool for detecting vulnerabilities of smart contracts[10][11]. Therefore, we chose Slither to annotate the vulnerability information of the dataset. The Smartbugs dataset contains 143 smart contracts that have vulnerability information. To construct the vulnerability dataset, we manually selected 20 smart contracts from this dataset, include 5 reentrant vulnerabilities, 5 timestamp-dependent vulnerabilities, 5 access control vulnerabilities, and 5 unchecked CALL return value vulnerabilities.

B. Evaluation metrics

We utilize standard metrics, include precision, recall, and \( F_1 \) to evaluate the effectiveness of OC-Detector. Additionally, we utilize Silhouette Coefficient[12] to evaluates the clustering effect of the K-means algorithm[13]. These evaluation metrics are frequently utilized in vulnerability detection approaches[14][15] and clustering algorithms[16].

C. Parameter settings.

To determine proper values of the parameters, we randomly select 500 contracts from the Xblock dataset and applying various vulnerability detection approaches. We measured the True Positive Rate and False Negative Rate of each approach by different similarity thresholds. Through these experiments, the optimal performance of OC-Detector, DC-Hunter, OC-Detector*, and OC-Detector w/o clustering are achieved with threshold values of 0.40, 0.50, 0.40, and 0.50, respectively. In subsequent experiments, we evaluated the vulnerability detection performance of each approach using these threshold values.

To determine the optimal number of clusters for clustering opcode instructions with different dimensions, we trained Word2Vec models to vectorize opcode instructions into different dimensions, and clustered the opcode instruction vectors with different dimensions. We calculated Silhouette Coefficient to determine the optimal number of clusters. The experimental results demonstrate that for opcode instruction vectors with 25 dimensions, the optimal number of clusters is 4, while for opcode instruction vectors with 256 dimensions, the optimal number of clusters is 20.

After that, we randomly selected 500 contracts from the Xblock dataset to detect potential vulnerabilities. The results indicate that the optimal vulnerability detection performance was achieved when opcode instructions were clustered into 4 clusters, Thus, the opcode instructions will be clustered into 4 clusters in subsequent experiments to maximize the effectiveness of detecting vulnerabilities.

D. Experimental results and analysis.

1) RQ1: How does the performance of OC-Detector compare to other existing tools?

To answer this question, we compare the vulnerability detection performance with other approaches, such as DC-Hunter, Security, SmartCheck, and Osiris. The experimental results are shown in Table I. As the result show, OC-Detector outperforms other four approaches in terms of precision, recall, \( F_1 \), and average time taken to analysis each contract. Specifically, the precision, recall, and \( F_1 \) value are improved from 0.05 to 0.36, from 0.02 to 0.43, and from 0.04 to 0.40, respectively. In addition, the average time to analysis one smart contract is reduced from 3.85 to 68.55 seconds, when compared to other four approaches. To validate the effectiveness of OC-Detector in solving the issue of false positives and false negatives caused by inconsistent opcode instructions generated by different versions of Solidity compilers, OC-Detector is compared with DC-Hunter. Since the contracts in the vulnerability dataset are compiled with compilers of version 0.4.25, 0.5.15, 0.7.1, and 0.8.7, we randomly selected 500 smart contracts compiled with compilers other than those above versions from the dataset. Then, these 500 smart contracts are analyzed by OC-Detector and DC-Hunter to detect vulnerabilities. The experimental results are shown in Table II.

As the result show, OC-Detector outperforms DC-Hunter in terms of precision, recall, and \( F_1 \). The precision, recall, and \( F_1 \) value are improved by 0.14, 0.27, and 0.27, respectively. Furthermore, the average time to analysis one smart contract is reduced by 3.87 seconds.

Answer to RQ1: OC-Detector outperforms other tools, include DC-Hunter, Security, SmartCheck, and Osiris, in terms of identifying vulnerabilities in smart contracts. Moreover,

<table>
<thead>
<tr>
<th>Approach</th>
<th>Avg Time(s)</th>
<th>Precision</th>
<th>Recall</th>
<th>( F_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC-Hunter</td>
<td>6.16</td>
<td>0.44</td>
<td>0.10</td>
<td>0.16</td>
</tr>
<tr>
<td>Security</td>
<td>23.79</td>
<td>0.46</td>
<td>0.23</td>
<td>0.31</td>
</tr>
<tr>
<td>SmartCheck</td>
<td>70.86</td>
<td>0.54</td>
<td>0.51</td>
<td>0.52</td>
</tr>
<tr>
<td>Osiris</td>
<td>21.18</td>
<td>0.23</td>
<td>0.14</td>
<td>0.17</td>
</tr>
</tbody>
</table>

OC-Detector effectively alleviates the problem of false positives and false negatives caused by inconsistent opcode instructions generated by different versions of compilers.

2) RQ2: How does the Word2Vec model affect the performance of OC-Detector?

To answer this question, OC-Detector is compared with OC-Detector* which utilize the glove.twitter.27B.25d\(^9\) model to vectorize opcode instructions instead of the Word2Vec model trained by using smart contract opcode instruction sequences, then calculate the similarity between the target contract and the contract of vulnerability dataset to detect vulnerabilities.

To determine the optimal number of clusters for the opcode instruction vectors generated by the glove.twitter.27B.25d model, we evaluate the results of clustering with different numbers of clusters by using Silhouette Coefficient. Experimental results show that the optimal number of clusters is 33.

The experimental results are shown in Table III. As the result shows, compared to OC-Detector*, the values of precision, recall, and \(F_1\) value are improved by 0.18, 0.08, and 0.13, respectively.

**Answer to RQ2:** Smart contract opcode instruction sequences can be used to learn context features of opcode instructions, which improve the performance of smart contract vulnerability detection.

3) RQ3: How does the clustering opcode instructions affect the performance of OC-Detector?

To answer this question, OC-Detector is compared with OC-Detector w/o clustering, which directly computed the similarity between the target contract and the contract in vulnerability dataset without clustering the opcode instructions. The experimental results are shown in Table IV.

As the result show, compared to OC-Detector w/o clustering, the precision, recall, and \(F_1\) value are improved by 0.17, 0.12, and 0.15, respectively.

**Answer to RQ3:** The clustering algorithm used by OC-Detector helps eliminate differences in opcode instructions compiled by different versions of compilers and improve the performance of detecting smart contract vulnerabilities.

\(^9\)glove.twitter.27B.25d: https://github.com/stanfordnlp/GloVe

### TABLE II: The performance comparison of OC-Detector and DC-Hunter with different compilers.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Avg Time(s)</th>
<th>Precision</th>
<th>Recall</th>
<th>(F_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC-Hunter</td>
<td>6.25</td>
<td>0.45</td>
<td>0.13</td>
<td>0.20</td>
</tr>
<tr>
<td>OC-Detector</td>
<td>2.38</td>
<td>0.59</td>
<td>0.40</td>
<td>0.47</td>
</tr>
</tbody>
</table>

### TABLE III: The performance comparison of OC-Detector and OC-Detector*.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Precision</th>
<th>Recall</th>
<th>(F_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OC-Detector</td>
<td>0.59</td>
<td>0.53</td>
<td>0.56</td>
</tr>
<tr>
<td>OC-Detector*</td>
<td>0.41</td>
<td>0.45</td>
<td>0.43</td>
</tr>
</tbody>
</table>

### TABLE IV: The performance comparison of OC-Detector and OC-Detector w/o clustering.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Precision</th>
<th>Recall</th>
<th>(F_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OC-Detector</td>
<td>0.59</td>
<td>0.53</td>
<td>0.56</td>
</tr>
<tr>
<td>OC-Detector w/o clustering</td>
<td>0.42</td>
<td>0.41</td>
<td>0.41</td>
</tr>
</tbody>
</table>

### V. Threats to Validity

External validity refers to ensure the generality of the experimental results. The contract to be checked in the experiment are taken from Xblock, and the contract in the Xblock dataset are real contract from the Ethernet, which being able to better validate the effectiveness of the approach. The smart contracts in the Smartbugs dataset are used to build the vulnerability dataset. These two datasets are widely used in studies of smart contract vulnerability detection\(^7\)[\(^8\)]. Although the vulnerabilities contained in these two datasets are representative, same performance cannot be ensured when applying OC-Detector on other datasets.

Internal validity is primarily related to factors that affect the correctness of the experiment. In this paper, we implemented the clustering algorithm using the Sklearn\(^10\), trained the Word2vec model using the Genimn\(^11\) and calculated similarity using the Levenshtein library to ensure the correctness of the implementation. Additionally, since DC-Hunter is not open-source, so we reimplemented it according to the descriptions of the paper. Although we have tested and checked results multiple times, the implementation details may differ from the original paper.

Construct validity is primarily concerned with the evaluation metrics used in the experiment. Precision, recall, and \(F_1\) value are used to evaluate the performance of detecting smart contract vulnerabilities in the experiment. These evaluation metrics are widely used in smart contract vulnerability detection approaches\(^17\)[\(^18\)]. In addition, Silhouette Coefficient is used to evaluate the effect of clustering, which is commonly used to in previous studies\(^19\)[\(^20\)].

### VI. Related Work

The machine learning-based vulnerability technology has been widely used in smart contracts. Previous research has focused on learning contract semantics and syntax using natural language processing techniques. For example, Wang et al.[\(^21\)] learned syntax and semantic features from bytecode to detect smart contract vulnerabilities, while Liu et al.[\(^22\)] learned code features from contract source code to detect vulnerabilities. However, extraneous code generates noise, which affects the performance of learning contract syntax and semantic information.

To solve this problem, some researchers proposed slicing technology to filter key code and reduce interference of code unrelated to vulnerabilities. For example, Han et al.[\(^3\)]

\(^10\) Sklearn: https://github.com/automl/auto-sklearn
\(^11\) Genimn: https://github.com/RaRe-Technologies/gensim
reduce noise by slicing and use graph embedding algorithms to convert program-dependent graphs into vectors to detect vulnerabilities. This approach has demonstrated that slicing can improve the performance of vulnerability detection.

Recent research has focused on using features learned through natural language processing techniques to detect vulnerabilities by machine learning models. For example, Yang et al. [23] proposed a new self-supervised learning approach for smart contract representation. Cai et al. [24] encode smart contract’s function into a graph with sufficient semantic features. They then utilize bidirectional gated graph neural network with a hybrid attention pooling layer to learn the code features, efficiently capturing vulnerability-related features from the graph for vulnerability detection. These approaches have been proven to be effectively to improve the performance of smart contract vulnerability detection.

To improve performance in detecting smart contract vulnerabilities, we learn opcode instruction features after slicing opcode instruction sequences and use clustering algorithms in machine learning to eliminate the differences between different opcode instructions to detect contract vulnerabilities.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a smart contract vulnerability detection approach based on clustering opcode instructions and implement a prototype tool OC-Detector. Experimental results on the Xblock dataset show that OC-Detector outperforms DC-Hunter, Security, SmartCheck, and Osiris in detecting smart contract vulnerabilities. Specially, mitigates false positive and false negative caused by inconsistent opcode instructions generated by different versions of compilers.

With the wide application of smart contracts, in addition to Solidity, Go, C++, and other programming languages are also commonly used for developing smart contracts, which also affected by typical vulnerabilities such as reentry, overflow and unchecked CALL return value. However, different smart contract programming languages inevitably generate different opcode instructions, which invalidate vulnerability detection approaches based on specific programming languages. As part of our future work, we plan to solve the issue of detecting vulnerabilities of smart contracts written in different programming languages.

ACKNOWLEDGEMENT

This work was supported in part by the Beijing Information Science and Technology University “Qin-Xin Talent” Cultivation Project (No. QXTCP C201906).

REFERENCES


Formalization and Verification of Data Auction Mechanism Based on Smart Contract Using CSP

Yingjia Du¹,*, Yuan Fei²*, Sini Chen¹, Huibiao Zhu¹
¹Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
²College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai, China

Abstract—Nowadays, the utilization of online auction platforms is becoming increasingly prevalent. Online auction provides a common and practical way for global buyers to compete fairly. Nevertheless, the anonymous environment may bring collusion among entities with effects on results. Compared with traditional mechanisms which rely on third-party platforms, the data auction based on smart contract can create a decentralized environment to avoid the occurrence of collusion. Meanwhile, there exists few research on the verification of its reliability and safety which is worth investigating from the perspective of formal methods.

In this paper, we apply Process Algebra CSP in modeling the data auction communicating system among five key entities. In addition, we use Process Analysis Toolkit (PAT) to realize the mechanism and verify five crucial properties, including deadlock freedom, data reachability, data correctness, anti-collusion capability and data security. The verification results indicate that the architecture of data auction based on smart contract can satisfy all the above requirements. Especially, the design of asymmetric encryption for the fundamental information ensures the non-occurrence of collusion in the auction. Additionally, the digital signature generated by private key attached to the message guarantees the safety of the interaction.

Index Terms—Data Auction Mechanism; Smart Contract; Process Algebra CSP; Modeling; Verification

I. INTRODUCTION

In the era of information, the demand for high-quality data, resistant to conventional pricing methods [1] has brought enormous popularity and prosperous landscape in the field of data auction.

The online data auction mechanisms have been proliferating [2]. Despite that, existing mechanisms, which excessively rely on third-party platform, can lead to information leakage [3], challenges of distrust between entities and other issues urgently to be addressed.

Meanwhile, the blockchain technology has obtained growing attention [4] as a promising solution for secured and traceable record which can be applied in the auction process. Xiong et al. [5] proposed the decentralized anti-collusion data auction mechanism based on smart contract, which enables secured and immutable record of the process [6] and invokes automated condition evaluation program.

As demonstrated in Fig.1, the mechanism based on smart contract is decentralized in data exchange. The smart contract replaces the role of auctioneers and result announcement in the traditional system [7]. Data providers can upload data to the smart contract and potential buyers can make confidential bids. Once the auction is over, smart contract will make the result public to participants and the buyer with the highest bid will gain ownership of the auctioned data. Details of the transaction are recorded and traceable [8]. Additionally, anyone involved in the auction can access the results. Thus, security and truthfulness of the data auction mechanism based on smart contract are crucial and formal methods can provide a systematic and rigorous approach to analyzing it.

Fig. 1. Data Auction Mechanism Based on Smart Contract

In this paper, we formalize the model of the anti-collusion data auction mechanism [5] based on smart contract utilizing Communicating Sequential Process (CSP) [9]. Besides, with the aid of Process Analysis Toolkit (PAT) [10], we verify the following crucial properties, including Deadlock Freedom, Data Reachability, Data Correctness, Anti-collusion Capability. Meanwhile, we simulate the scenario with the presence of an intruder and establish a model to express its behavior of decrypting and faking messages. We can infer from the verification results that the mechanism is stable and reliable.

The remainder of the paper is organized as follows. We deliver a brief introduction to the architecture of the data auction and the basic knowledge of CSP in Section II. The subject of Section III is the formalized model of the system. Moreover, Section IV presents verification process and results. We discuss the conclusion and future work in Section V.

II. BACKGROUND

In this section, we start with an overview of the proposed data auction system based on smart contract by Xiong et al. [5]. We also give a brief introduction to process algebra CSP.
A. Data Auction Mechanism Architecture

As demonstrated in Fig.2, the framework of the mechanism based on smart contract is decentralized to improve the efficiency and is comprised of five entities as below.

![Image of the Data Auction Framework]

**Fig. 2. The Data Auction Framework**

- **Data Seller (DS):** It owns the data and the revenue.
- **Data Buyer (DB):** It intends to acquire auctioned data.
- **Data Trading Centre (DTC):** It is the institution where copies of transaction can be properly stored.
- **File System (FS):** It provides safe storage for the data.
- **Smart Contract (SC):** It plays a crucial role in recording the information and managing the transaction.

The data auction system based on smart contract adopts a single-round sealed-bid auction architecture. Only one data seller and multiple data buyers are allowed to take part in the same auction. The auction process can be divided into three steps: Preparation, Process and Conclusion, which are illustrated in Fig.3. The definitions of key variables and functions are given as follows.

- **Sealed Price** \( sp_i \): It is the price proposed by the seller as the minimum transaction price and \( i \) stands for its id.
- **Sealed Bid** \( sb_j \): The variable shows true bid of the buyer sealed in the process and \( j \) stands for the buyer’s id.
- **Bid Mask** \( bm_j \): It is required to be larger than the corresponding \( sb \) and is public to disguise the true bid.
- **paySDeposit()** The function records the information of the seller and deduct a deposit from its account.
- **payBDeposit()** It is activated when a buyer is willing to take part in the auction. Moreover, it depicts the behaviour of paying deposit to get the admission to the auction.
- **dataAuction()** It checks whether the transaction meets the condition \( sp_i \leq sb_j \leq bm_j \).
- **payBid()** This function withdraws corresponding bid price from the account of the winning buyers.
- **verifyInfo()** It is used to check whether a certain buyer is on the winning buyers’ list to authenticate its identity.

Next, we will present detailed steps of the data auction.

1. The Data Seller sends the reserved price \( sp_i \) to the Smart Contract with some supplementary basic information.
2. The Data Seller informs the Data Trading Centre and Data Buyer of the auctioned data information.
3. The Data Buyer decides whether to participate in the current auction. If the answer is yes, it pays the deposit to the Smart Contract. Otherwise, it quits.
4. The Data Seller and Data Trading Centre receive the list of participants in the auction from the Smart Contract.
5. The qualified Data Buyer \( j \) submits a sealed bid \( sb_j \), which is based on evaluation of the auctioned data to Smart Contract and broadcasts a bid mask \( bm_j \), which is a disguise of the true bid.
6. The Smart Contract forwards the same bid information to the Data Trading Centre.
7. The Smart Contract executes the auction program and make the highest Data Buyer accessible to the auctioned data.
8. The Smart Contract broadcasts the list of winner(s) to the Data Seller, the Data Trading Centre and all Data Buyer.
9. The winning buyer pays for the bid to the Smart Contract.
10. The Smart Contract confirms that it has received the currency from the Data Buyer.
11. The Smart Contract generates a token randomly.
12. The Smart Contract provides the winning Data Buyer with a created token to download the auctioned data.
13. The Data Buyer shows its id for File System to validate.
14. The File System downloads the auctioned data by token.
15. Data is accessible to the winning Data Buyer.
16. The File System sends confirmation to the Data Trading Centre and Data Seller.

![Image of the Data Auction Flow Chart]

**Fig. 3. The Data Auction Flow Chart (adapted from [5])**

B. CSP

Communicating Sequential Processes (CSP) is a representative process algebra based on logic [9]. The definition of the syntax of CSP used in this paper is given as below.

\[
P, Q ::= SKIP | c?u \to P | clv \to P | P \parallel Q | P \parallel [X]Q | P < B > Q
\]

- **SKIP** stands for that the process which does nothing, but terminates immediately.
- \( c?u \to P \) denotes that the process receives a value through the channel \( c \), assigns it to variable \( u \) and behaves as \( P \).
- \( clv \to P \) indicates that the process sends message \( v \) through the channel \( c \). Then it performs as process \( P \).
• $P \sqsubseteq Q$ represents that the process behaves like either process $P$ or $Q$, determined entirely by the environment.
• $P||Q$ refers to the concurrent execution of $P$ and $Q$. Same events in a common alphabet require synchronization.
• $P||[X]|Q$ describes the concurrent execution of $P$ and $Q$ on the set of channels $X$.
• $P\llcorner b\lrcorner Q$ stands for the condition. When Boolean variable $b$ is true, it will perform as $P$. Otherwise, it will behave like $Q$.

### III. MODELING

In this section, we give the formalized model of the system of data auction [5] using CSP. We begin by the definition of sets, messages and channels. Then, we formalize the architecture described in Section II by modeling entities respectively.

#### A. Sets, Messages and Channels

The whole architecture is rather complicated. Thus, we divide the involved elements into sets as the basic components of the model according to their types and functionalities. We give an introduction to the sets as below.

- **Entity** involves Data Seller (DS), Data Buyer (DB), Data Trading Centre (DTC), File System (FS) and Smart Contract (SC) mentioned in Fig.2.
- **ID** defines serial number of DB and DS. $ID = BID \cup SID$.
- **Account** consists DB and DS’s funding. Account = BAccount $\cup$ SAccount.
- **Price** includes sealed price $sp$ of DS, sealed bid $sb$ and bid mask $bm$ of DB. Price = SPrice $\cup$ SBid $\cup$ BMask.
- **Key** contains public and private keys belonging to Entity. $Key = k_{pub} \cup k_{pri}$

$k_{pub} = \{ bk_{pub}, sk_{pub}, sck_{pub}, f_{k_{pub}}, t_{k_{pub}} \}$

$k_{pri} = \{ bk_{pri}, sk_{pri}, sck_{pri}, f_{k_{pri}}, t_{k_{pri}} \}$

- **Sig** is composed of the identification of Entity. $Sig = \{ bsig, ssig, scesig, f_{sig}, tsig \}$

- **Par** is composed of DB’s participations.

In addition, $E$ and $D$ describe message encryption and decryption while $G$ and $V$ refer to digital signature generation and verification using $k_{pub}, k_{pri}$ in blockchain.

- $E (k_{pub}, msg)$ signifies $k_{pub}$ is used to encrypt $msg$.

- $D (k_{pri}, E (k_{pub}, msg))$ shows that $k_{pri}$ is able to decrypt $msg$ encrypted by corresponding $k_{pub}$.

- $G (k_{pri}, sig)$ denotes $k_{pri}$ can be used for generating $sig$.

- $V (k_{pub}, E (k_{pri}, sig))$ shows that $k_{pub}$ can be utilized to recognize identification $sig$ generated by $k_{pri}$.

Moreover, we classify and abstract the messages transferred between entities as tuples to reduce redundancy.

$Tuple_{e1} = \{ sid, sname, stype, slink \}$

$Tuple_{e2} = \{ bid, par \}$

$Tuple_{e3} = \{ bid, baddress, bn \}$

$Tuple_{e4} = \{ bid, baddress, baccount, par \}$

$MSG_{req} = \{ \{ M_{req1}, Tuple_{e1}, G (k_{1}, sig) \}, M_{req2}, Tuple_{e2}, G (k_{2}, sig) = E (k_{3}, sp) \}_{id \in SID, sname \in Name, stype \in Type, slink \in Link, sp \in Price, saccount \in SAccount}$

$\begin{align*}
& \text{e.g. } M_{req1} = \text{sid, sname, stype, slink, } G (k_{pri}, sig) \\
& \text{Similarly, we omit the remaining conversion process.} \\
& MSG_{req} = \{ \{ M_{req1}, Tuple_{e1}, G (k_{1}, sig) \}, M_{req2}, Tuple_{e2}, G (k_{2}, sig) = E (k_{3}, sp) \}_{id \in SID, sname \in Name, stype \in Type, slink \in Link, sp \in Price, saccount \in SAccount} \\
& \text{show that } G \text{ of the proposed mechanism without interception and forgery. We then take the situation in which intruder exists into account to establish a complete model. Due to the page limitation, we omit the modeling process of DataTradingCentre and FileSystem.} \\
& \text{As illustrated in Fig.3, the system has three stages: Preparation, Process and Conclusion executed sequentially. Furthermore, in accordance with the base of data auction, the message transmission process needs to comply with rules of blockchain. The sender should encrypt message with } k_{pri} \text{ and generate}
\end{align*}$
a digital signature by $G(k_{pri}, sig)$ as mentioned in Part A. Similarly, the receiver can call \texttt{verifySig} $(k_{pub}, E(k_{pri}, sig))$ to verify the sender’s identity by $k_{pub}$.

**System** $=_{df} DataBuyer \mid DataSeller \mid FileSystem \mid SmartContract \mid TradingCentre$

**System** $=_{df} System_{0}[Intruder\_Path] | Intruder$

### C. DataSeller Modeling

DataSeller is responsible for providing data information and proposing the sealed price $sp$ as the reserved auction price which is not exposed to others until the auction ends.

**DataSeller** $=_{df}$

\[
\begin{align*}
& (encryptInfo(sck_{pub}, sp) \rightarrow ComDSS!Msg_{req2} \rightarrow \\
& paySDeposit(sid) \rightarrow ComDST!Msg_{req1} \rightarrow \\
& ComDBDS!Msg_{req} \rightarrow ComDSS?Msg_{req1} \rightarrow \\
& verifySig(sck_{pub}, G(sck_{pri}, sig)) \rightarrow DataSeller \\
& <(sig == sssig) > fail \rightarrow DataSeller
\end{align*}
\]

\(\langle (state == Preperation) >\)

\[
\begin{align*}
& ComDSS?Msg_{req} \rightarrow \\
& verifySig(sck_{pub}, G(sck_{pri}, sig)) \rightarrow \\
& (ComDSS?Msg_{ack} \rightarrow DataSeller) \\
& <(sig == sssig) > fail \rightarrow DataSeller
\end{align*}
\]

\(\langle (state == Conclusion) > fail \rightarrow DataSeller\rangle\)

In the preparation stage, we define $encryptInfo(k, sp)$ to disguise the lowest price. $paySDeposit(sid)$ is designed to deduct the deposit from the DataSeller’s account to avoid bought-in. We express the behaviour of delivering basic information $Msg_{req1}$, $Msg_{req2}$ to SmartContract and DataTradingCentre respectively. $verifySig(k, sig)$ returns the decrypted sig using the corresponding public key. We compare the result with the original sig from the expected SmartContract to check the sender’s identity. If the result is true, the system performs smoothly. Otherwise, the identity verification fails and the transaction terminates.

The variable $par$ is defined to record the participation of DataBuyer. In the conclusion step, DataSeller is informed of the winner from SmartContract and obtains the revenue as Step 8 in Fig.3. The specific contents are given as below.

\[
\begin{align*}
& Msg_{req1} = [\text{bid, account, par, } G(bk_{pri}, sig)] \\
& Msg_{req2} = \text{Tuple}(\text{bid, account, par, } G(sck_{pub}, sp)) \\
& Msg_{req1} = \text{[bid, account, par, } G(sck_{pri}, sig)].
\end{align*}
\]

### D. DataBuyer Modeling

DataBuyer is the crucial component of the data auction architecture. The set of buyers makes decisions on whether to take turns to participate and make public bid mask $bm$ to hide true bid $sb$. We firstly model the behaviour of a single DataBuyer and integrate multiple buyers as a whole. The variable $DB$ stands for the number of participants.

In the first period, in Step 2 in Fig.3, DataBuyer receives $Msg_{req1}$ from DataSeller and checks its signature to confirm that the message is from the specified sender. DataBuyer decides individual participation which is defined by the variable $par$. If the value equals to true, DataBuyer shows its willingness to make bid on the data. Then, it pays for the required deposit. In the second step, DataBuyer takes part in broadcasting $bm$ and applies $encryptInfo(k, sb)$ in disguising $sb$ from entities except for the SmartContract in Step 5. Finally, it gets the result of the data auction in $Msg_{req4}$. We define the comparison between DataBuyer’s id with the winner’s as a condition statement. If it is true, winning buyer pays for the bid according to $sb$ and receives the token to download data encrypted by its public key. $decryptInfo(k, token)$ serves to obtain the token generated by SmartContract and encrypts it to FileSystem to obtain the data in Steps 11-13. Winning DataBuyer receives $Msg_{data}$ in the end if the result of verification is true. $Msg_{req4} = [\text{bid, address, account, par, G(bk_{pri}, sig)}]$. $Msg_{req1} = [G(sck_{pri}, sig), E(bk_{pub}, token)]$.

\[
\begin{align*}
& DataBuyer =_{df} \langle \text{encDec} \rangle \langle \text{com}\_{mk}\rangle \langle \text{com}\_{sk}\rangle \\
& <(state == Preparation) >\rangle
\end{align*}
\]

\[
\begin{align*}
& encryptInfo(sck_{pub}, sb) \rightarrow \\
& ComDBS?Msg_{req2} \rightarrow DataBuyer \\
& <(state == Process) >\rangle
\end{align*}
\]

\[
\begin{align*}
& ComDBS?Msg_{req} \rightarrow \\
& verifySig(sck_{pub}, G(sck_{pri}, sig)) \rightarrow \\
& payBid() \rightarrow ComDSS?Msg_{ack} \rightarrow \\
& ComDBS?Msg_{ack} \rightarrow \\
& verifySig(sck_{pub}, G(sck_{pri}, sig)) \rightarrow \\
& decryptInfo(fk_{pub}, msg_{1}) \rightarrow \\
& ComDBF?Msg_{data, msg_{2}} \rightarrow \\
& verifySig(fk_{pub}, msg_{2}) \rightarrow \\
& DataBuyer \langle (msg == data) > \rightarrow DataBuyer \\
& <(state == Conclusion) > DataBuyer \\
& \text{Skip} \rightarrow DataBuyer
\end{align*}
\]

### E. SmartContract Modeling

SmartContract is the basis of the mechanism. It defines rules for other entities to follow, selects winning buyer(s) and broadcasts the list of winner(s).

At the beginning, as Step 1 in Fig.3 denoted, DataSeller sends $Msg_{req2}$ to inform it of the overview of the auctioned data. SmartContract then conducts validation of the identity of sender in $verifySig(k, sig)$. It collects the deposit and shows the auction participation to DataSeller.
In the Process stage, it gets the specific data of bid in $Ms_{req\_q21}$. It also forwards $Ms_{req\_q22}$ to DataTradingCentre. Then, it decrypts the message with its private key in $decryptInfo(k, msg)$. It calls the $data\_Auction()$ program. $Ms_{req\_q22} = \{bid.address.bm.G(sck_{pri}, sig), E(sck_{pub}, sb)\}$.

$$data\_Auction(bid, baccount, bm, sp, sb, par) = df$$

$$\begin{align*}
&\text{winner.append(bid)} \\
&\langle sb == \text{high\_bid} \rangle \triangleright withdrawBDeposit(bid) \\
&\langle \text{winner.empty()} \rangle \triangleright \text{winner.append(bid)} \\
&\langle (sb > \text{high\_bid}) \rangle \triangleright withdrawBDeposit(bid) \\
&\langle \text{condition} \rangle \triangleright \text{withdrawBDeposit(bid)}
\end{align*}$$

condition : \( \text{par} == \text{true} \land \text{sb} \leq \text{baccount} \land \text{sp} \leq \text{sb} \leq \text{bm} \)

The function $data\_Auction(bid, baccount, bm, sp, sb, par)$ checks whether the decrypted $sb$ meets the rules and compares it with the recent highest bid $\text{high\_bid}$.

At last, in Step 8 in Fig.3, it broadcasts winning buyers’ id. When it confirms the payment in $Ms_{ack}$, it randomly generates a token and encrypts it to ensure confidentiality.

SmartContract’ $= df$

$$\begin{align*}
&\text{ComDSS}\_\!Ms_{req\_q21} \rightarrow \\
&\text{verifySig}(sk_{pub}, G(sk_{pri}, sig)) \rightarrow \\
&\text{ComDSS}\_\!Ms_{req\_q23} \rightarrow \\
&\text{decryptInfo}(sk_{pri}, E(sck_{pub}, sp)) \\
&\text{decryptInfo}(sk_{pri}, E(sck_{pub}, sb)) \\
&\text{data\_Auction}() \rightarrow \\
&\langle sig == \text{ssig} \rangle \triangleright \text{fail} \rightarrow \text{SmartContract’}
\end{align*}$$

$\langle \text{state} == \text{Preparation}\rangle$:

$$\begin{align*}
&\text{ComDSS}\_\!Ms_{req\_q21} \rightarrow \\
&\text{verifySig}(sk_{pub}, G(bk_{pri}, sig)) \rightarrow \\
&\text{ComTSIMs}_{r\_eq\_q1} \rightarrow \\
&\text{decryptInfo}(sk_{pri}, E(bk_{pri}, sig)) \\
&\text{data\_Auction}() \rightarrow \\
&\langle sig == \text{bsig} \rangle \triangleright \text{fail} \rightarrow \text{SmartContract’}
\end{align*}$$

$\langle \text{state} == \text{Process}\rangle$:

$$\begin{align*}
&\text{ComDSS}\_\!Ms_{req\_q23}.bid.G(sck_{pri}, sig) \rightarrow \\
&\text{ComDSS}\_\!Ms_{req\_q23}.bid.G(sck_{pri}, sig) \rightarrow \\
&\text{ComTSIMs}_{r\_eq\_q2}.bid.G(sck_{pri}, sig) \rightarrow \\
&\text{ComDSS}\_\!Ms_{req\_ack} \rightarrow \text{generateToken()} \rightarrow \\
&\text{EncryptInfo}(bk_{pub}, \text{token}) \rightarrow \\
&\langle \text{ComDSS}\_\!Ms_{req\_r} \rightarrow \text{SmartContract’}\rangle \\
&\langle \text{bid} \in \text{winner} \rangle \triangleright \text{Skip} \rightarrow \text{SmartContract’}
\end{align*}$$

$\langle \text{state} == \text{Conclusion} \rangle \triangleright \text{fail} \rightarrow \text{SmartContract’}$

SmartContract $= df$ SmartContract’[

ComDSS\_\!\{ComDSS\} $\leftarrow$ ComDSS\_\!\{ComDSS\},
ComDSS\_\!\{ComDSS\} $\leftarrow$ FakeSC\_\!\{FakeSC\}].

F. Intruder Modeling

The intruder causes unexpected accidents by eavesdropping and forging messages. We define a set Fact including facts it might obtain from the interaction between truthful entities.

$$\text{Fact} = df \{ \text{bid, address, bm, G(bk_{pri}, sig)}, E(sck_{pub}, sb) \}$$

$\text{bid} \in \text{BID}, \text{baddress} \in \text{BAddress}, \text{bm, sb} \in \text{Price},$

$\text{bk}_{pri} \in \text{b_{pri}}, \text{sig} \in \text{Sig(sck_{pub} \in \text{Pub})} \cup \text{Entity}.$

The intruder can deduce new facts from the accessible Fact. We describe the progress of deducing content of fact $f$ from the origin fact $F$ by the symbol $F \rightarrow f$.

We can learn from the first rule that the intruder can decrypt messages in asymmetric encryption.

The second rule shows that when set $F$ is the subset of $F'$, intruder can deduce $f$ not only from $F$ but also from $F'$.

$\{k_{pri}, E(k_{pub}, c)\} \rightarrow c \quad (2) F \rightarrow f \\ F \subseteq F' \Rightarrow F' \rightarrow f$

We define the function $getInfo(msg)$ to imply the knowledge which the intruder intercepts from the process.

$\text{getInfo}(msg) = df \{ bid, address, bm, G(bk_{pri}, sig), E(sck_{pub}, sb) \}$

We define a channel $Deduce$ to deduce new facts:

Channel Deduce : Fact(Fact)

Then, we give a formalized model of Intruder’ as follows.

Intruder’ $(F) = df$

$$\begin{align*}
&\text{msg} \in \text{MSG} \rightarrow \text{Intruder’}(F \cup \text{getInfo}(msg)) \\
&\text{msg} \in \text{MSG} \rightarrow \text{Intruder’}(F \cup \text{FakeSC}\_\!msg) \\
&\text{new} \in \text{Fact}, \text{msg} \in F, F \cup \text{new} \rightarrow \text{Deduce.new.(F)} \\
&\langle \text{Data\_Faking\_Success} == \text{true} \rangle \rightarrow \\
&\langle \text{Intruder’}(F \cup \text{new}) \rangle
\end{align*}$$

The intruder adds $\text{getInfo}(msg)$ to its current knowledge when intercepting $msg$. It can decrypt $msg$ from DataBuyer and change the content of $msg$ to send it to SmartContract. If the $msg$ cannot be identified from an unauthorized user and is received, it means that the intruder successfully modifies the content of the message and completes an attack. Then, we assign true to $\text{faking\_flag}$. Now we give the model of Intruder. The parameter $BK$ is its basic knowledge.

$$\text{Intruder} = df \text{Intruder’}(BK), \quad BK = df \text{Entity} \cup \text{bk_{pub}} \cup \{sck_{pri}\}$$

IV. VERIFICATION AND RESULTS

In this section, we analyze and verify five properties of the constructed model in the model checker PAT [10] to evaluate the reliability of anti-collusion data auction mechanism.

A. Properties and Analysis

- **Deadlock Freedom:**

  Deadlock represents the situation when no entity can interact with each other normally in the system. PAT has a primitive to describe it.

  #assert System() deadlockfree;

- **Data Reachability:**

  Reachability ensures that all valid data can be transferred between subjects in the process until the end. We assign the defined Boolean variable $data\_reachable\_flag$ to true when the process obtains the expected target data.

  #define Data_Reachable reachable_flag == true; #assert System() reaches Data_Reachable;

- **Data Correctness:**

  Correctness describes the ability ensuring that the final displayed price is consistent with the price proposed by the buyer. It also indicates that the mechanism can draw the correct conclusion. We assign the defined Boolean
variable `data_correct_flag` to true when receiving the true result with no mistakes.

```c
#define Data_Correct data_correct_flag == true;
#define assert System() reaches Data_Correct;
```

- **Anti-collusion Capability:**
  
  *Collusion* stands for the situation in which data buyers can observe others’ prices to make adjustment to their own bid price to keep winning price as low as possible. We should avoid its occurrence to make the auction fair. In the supposed situation, the buyer intercepts the message in order to get sealed bid to influence the result. The variable `current_bid` represents the data transferred on the channel, which has the risk of being intercepted. Moreover, the variable `true_bid` stands for the true bid price from the corresponding buyer. We focus on comparing `current_bid` with `true_bid` to verify the property.

```c
#define Anti_Flag current_bid != true_bid;
#define assert System() reaches Anti_Flag;
```

- **Data Security:**
  
  *Security* refers to the situation where the intruder cannot forge data successfully. Supposing there exists an intruder, we define the Boolean variable `faking_flag` to record the effect of attack. We assign it to true when the intruder successfully changes the value and has influence on the auction result to check whether the system is safe.

```c
#define Data_Faking_Success faking_flag == true;
#define assert System() reaches Data_Faking_Success;
```

**B. Verification Results**

We use PAT to simulate the behaviour of the constructed model. Then, we verify the above properties. The invalid result of `Data_Faking_Success` equals to the confirmation of `Data Security`. The result is illustrated in Fig. 6, implying that the system can always satisfy five properties, including *Deadlock Freedom, Data Reachability, Data Correctness, Anti-collusion Capability* and *Data Security*.

![Fig. 6. Verification Results of the Model](image)

The verification results indicate that the auction mechanism can avoid deadlock. At the same time, it can obtain the target data from an authorized user and achieve proper result. Moreover, it can protect messages from interception and tampering in the process by encrypting bid price and generating digital signatures which intruder is unable to fabricate.

Furthermore, the system can provide users with a way of recording the origin data and maintain data copies permanently which cannot be modified by intruders owing to digital signature authentication. The intruder can decrypt sealed `sb` and modify it by acquired `sckpri` when the data leakage occurs. However, it cannot imitate the signature of the DataBuyer. As a result, SmartContract receives `msg` from Intruder and verifies its certification which `msg` cannot pass. Hence, the attack is invalid and has no effect on the auction result.

Thus, we can come to the conclusion that anti-collusion auction mechanism based on smart contract can guarantee high truthfulness, safety and reliability.

**V. CONCLUSION AND FUTURE WORK**

In this paper, we adopted process algebra CSP to model the communication process of this architecture of the proposed [5] decentralized data auction scheme. It is based on smart contract which can solve the problem of collusion. We verified five properties: *Deadlock Freedom, Data Reachability, Data Correctness, Anti-collusion Capability* and *Data Security* with the assistance of the model checker PAT. As demonstrated in the verification results, it satisfies all properties, indicating the distributed mechanism is of great reliability, confidentiality and security under any circumstances. It guarantees to return the valid result of the data auction by encrypting vital information. Besides, it creates digital signatures attached to the message transferred in the system to ensure the safety.

In the future, we will put forward analysis on security strategies of smart contract and take the system of multiple sellers into consideration. Meanwhile, we will also introduce more types of attack into authentication process, such as replay attack and reentrancy attack.

**VI. ACKNOWLEDGEMENTS**

This work was partially supported by the “Digital Silk Road” Shanghai International Joint Lab of Trustworthy Intelligent Software (No. 22510750100), 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**


Session DMA: Data Mining and Analytics
Abstract—Automated machine learning (AutoML) is an increasingly popular approach to building machine learning (ML) models without the need for extensive human intervention. One key component of AutoML is automated data ingestion, which involves automatically collecting, cleaning, and preparing data for use in ML models. This paper aims to analyze the literature in order to identify how automated data ingestion is being developed in the literature. To achieve this goal, a survey was conducted on the state-of-the-art of automated data ingestion using a method based on a systematic literature review, in order to identify the existing practices. A total of 12 articles were initially found, however, after applying filters, only six of them were ultimately utilized in the research, showing that visual data navigation and validation as well as metadata inference are important features for automated data ingestion focused in AutoML.

Index Terms—automl, automated data ingestion, machine learning

I. INTRODUCTION

Although data are important organizational assets, they are difficult to handle, requiring skilled professionals to process, and analyze them. However, most people do not have professional training in technology [1], thus generating the need for software that allow data mining and analysis in an automated way, called AutoML systems. Such systems abstract several operations inherent to ML, allowing users to perform data analysis activities in a faster and more accessible way [2].

The first step in machine learning is data ingestion, which has as its main goal to consistently bring data into the ML pipeline [3]. To achieve that, the ingested data needs to be handled in a way that it can be consumed in the next steps of the flow. Since the main task of the user in an AutoML system is, according to F. Hutter, L. Kotthoff and J. Vanschoren [2], to provide the data to the flow, Data Ingestion becomes a key system component.

Automated data ingestion is particularly important in machine learning because it can help to reduce the time and effort required to prepare data for modeling. By automating the process of data wrangling, ML practitioners can spend more time building and refining their models. This can lead to faster and more accurate models, as well as more efficient teams [4].

With the motivation to identify the ideal way to perform the data ingestion step for AutoML systems, this work analyzes the literature to obtain information about methods that best suit this purpose. With the information gathered, this paper seeks to list the practices and techniques used in the articles analyzed that fit the needs of AutoML.

II. BACKGROUND

A. AutoML

Machine learning can be defined as an area of knowledge that is designed to bring to computers the ability to learn without the need for explicit programming [5]. As described by H. Hapke and C. Nelson [3], the ML model pipeline can be summarized into nine well-defined steps, starting at data ingestion and generating a loop that restarts upon reaching model feedback.

AutoML is a broad term, generally used to define systems that seek to automate ML activities. Its fundamental motivation for existence is due to some of the main limitations of the development of activities related to ML, being the need for a professional specialized in the development of such algorithms and the need for domain knowledge linked to the subject on which the algorithm is based, besides the fact that some related activities eventually become tedious and repetitive. AutoML tools can help accelerate the development of ML models, and democratize access to ML for individuals who do not have extensive expertise in data science [6].

Although automation of ML activities can be developed for any of its steps, according to F. Hutter, L. Kotthoff and J. Vanschoren [2] the most basic tasks for automation are hyperparameter optimization (HPO), meta-learning and neural architecture search. With the automation of such activities the authors indicate that several advantages are generated, such as: reduction of human effort in the development of data mining activities, improved performance of algorithms, improved reproducibility of academic work, reuse of successful models, improved feature selection, etc.

B. Automated Data Ingestion

The data ingestion is characterized by obtaining and transporting data from an external source into the ML workflow. This process is typically used by organizations to streamline the data collection process and reduce the need for manual data entry, improve data quality and accuracy, and save time and resources, which can improve decision-making and reduce the risk of costly mistakes. The data ingestion is usually suitable for tabular data, commonly in CSV, XML or database tables [3].
Within the methods found in the literature for developing data ingestion, we can highlight batch and streaming. Batch data ingestion is usually done through ETL (extract, transform, and load) routines that collect data from an external source and bring it into the workflow. The batch technique is used for data that does not need to be consumed in real time, such as a database snapshot. Streaming data ingestion, on the other hand, is used in cases where there is a need for real-time data consumption and specific technologies are used to support this type of need [7].

The data ingestion process can be done through a variety of methods, including Application Programming Interfaces (APIs) and other data integration tools. This means that data can be transferred between systems in a structured format, reducing the risk of errors and ensuring data consistency. File transfer protocols, such as FTP and HTTP, are also commonly used for automated data ingestion, these protocols allow for the transfer of large files between systems, making it possible to collect data in bulk [8].

C. Related Work

This section shows secondary studies that sought to identify data ingestion methods for various purposes or related activities. The main difference of the present work to the others found in the literature is the focus on the analysis of methods relevant to AutoML systems during the search for data ingestion solutions, resulting on the search for automated data ingestion.

In paper [7] an analysis of the state-of-the-art of data ingestion and integration is performed, due to the wide range of existing tools and methods for these types of operations. Throughout the paper, the main features and technologies used are analyzed, dividing them into three main groups: data integration tools, stream data ingestion tools and cloud data processing tools. On the subject of data integration tools, it is indicated that usually the platforms have a graphical interface and are batch processing oriented, being represented mainly by Microsoft SQL Server Integration Services1, Talend 2 and Sqoop3. Regarding the Stream data ingestion tools, the authors present as main characteristics the support to data pipelining, support to streaming processing and a wide variety of designs and support for users, such as Apache tools4, Oracle Stream Analytics5 and IBM Streams6. Finally, regarding cloud-based tools, the main functionalities listed are the support for batch processing and streamlining with the ability to be applied both for local and cloud situations, highlighting in this context tools such as Azure Stream Analytics7, IBM Streams6 etc.

In [9] the authors list the data ingestion methods used in the literature and develop a tutorial, through a survey, for developing Hybrid Transactional/Analytical Processing (HTAP) solutions. The authors list two design options for HTAP: a single system for OLAP and OLTP that allows an organization for data ingestion in common databases and analytical processing in columnar databases, and the second option would be two separate systems for OLTP and OLAP, this way the data stored in the OLTP system would have the original data that would be ingested in the OLAP system through an ETL routine, keeping data stored in both formats, allowing both types of analysis, but there are alternatives where the data for both OLTP and OLAP are stored in the same database, even if they come from different systems.

[10] aim through a survey to review the state-of-the-art of recent big data solutions, to support the selection of the more appropriate technologies. Throughout the work, comparisons are made between the technologies present in the market for the most different layers of big data. In the context of data ingestion, the authors present that this technology is inserted in the data access layer and the main tools used for development are Apache Sqoop5, Apache Flume8 and Apache Chukwa9, characterized by stream data processing.

III. METHODOLOGY

For this study, a state-of-the-art survey methodology, based on the systematic literature review proposed by Kitchenham and Charters [11], was used. To achieve the research objectives, five steps were taken to plan the research, conduct exploratory research, and refine the research results. Each stage will be detailed in the following subsections.

A. Research Planning

In the planning stage were defined the research method, the general and specific objectives of this work, as well as the research questions. Considering the main focus of this work as mapping the methods used in the literature for automated data ingestion, the research questions were defined as follows:

- **RQ1**: How is automated data ingestion implemented?
- **RQ2**: What are the metrics observed for data ingestion performance?
- **RQ3**: Which are the most suitable techniques for automated data ingestion in AutoML systems?

Research question 1 (RQ1) aims to identify broadly what are the main features of automated data ingestion systems, as well as the technologies used to develop them. In research question 2 (RQ2), the goal is to identify how to quantitatively evaluate data ingestion methods, so that comparisons between them are possible. Finally, research question 3 (RQ3) seeks to compare the characteristics of existing automated data ingestion systems with the needs of an AutoML system.

B. Search Strategy

The following procedure was made iteratively to build the search string used to perform the state-of-the-art analysis:

2. https://www.talend.com/
1) Definition of terms that would serve as keywords based on the research questions and objectives of this paper;
2) Identification of synonyms for the defined keywords, aiming to increase the search results;
3) Use of Boolean operators such as “AND” and “OR” to filter the results;
4) Evaluation of the results obtained, by analyzing the titles of the papers listed as research results.

After the completion of the procedure mentioned above, it was defined as search string: (“automated data ingestion”). The string was used to obtain papers from the research sources, searching for journals and conference papers: IEEExplore, ACM, Scopus, Science Direct. As a result of this phase of the search, a total of 12 papers were obtained, one from IEEExplore, two from ACM, three from Scopus and six from Science Direct.

C. Study Selection

By reading the title and abstract of the articles, inclusion, and exclusion criteria were applied according to Table I. The application of such criteria allowed a reduction in the initial number of articles, leaving a total of six articles, of these: one from IEEExplore, zero from ACM, one from Scopus and four from Science Direct.

<table>
<thead>
<tr>
<th>Inclusion criteria</th>
<th>Exclusion criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conference, Proceedings or Journal papers</td>
<td>Papers that are not in English</td>
</tr>
<tr>
<td>Primary studies</td>
<td>Papers under 3 pages in length</td>
</tr>
<tr>
<td>Complete papers</td>
<td>Duplicate or similar papers</td>
</tr>
<tr>
<td>Papers published after 2017</td>
<td>Papers unavailable for download or viewing</td>
</tr>
<tr>
<td>Papers that respond at least two research questions</td>
<td></td>
</tr>
</tbody>
</table>

D. Quality Criteria Application

Aiming to filter the previously selected papers, for the present study, one quality criterion was chosen, namely: paper brings information about automated data ingestion. After applying the quality criteria, 6 articles were selected.

E. Evidence Collection

In the evidence collection stage, the selected papers were read, aiming to identify whether they could answer the research questions listed in the planning stage. A scoring system was developed for the level of responsiveness of each research question by the article, requiring a sum greater than 1.5 for the work to be considered relevant to the research. The scoring was defined as follows: answer completely (score: 1), partially answers (score: 0.5), do not answer (score: 0).

IV. DATA ANALYSIS AND SYNTHESIS

A. RQ1 – How is automated data ingestion implemented?

The work developed by Otamendi et al. [12] presents an automated data ingestion module that provides the user with the ability to manipulate geospatial data, images obtained from satellites and sensors. The user must ingest the first data manually and specify the features and characteristics of each image. To facilitate this step, the authors have developed data-driven methods to extract metadata from other databases and attempt to infer the metadata of the selection being ingested. In addition, the automated data ingestion module is capable of processing the data periodically after the ingestion is configured, and can capture data from satellites on premise or on cloud.

The paper authored by Hacker et al. [13] features a prototype, called Experiment Dashboard (ED), with the goal of demonstrating automated data ingestion capabilities for research lab data with a minimal amount of effort and expertise. The data ingester automatically captures the data at the source and ingests it in real time into the database, where the information is queried for the ED. The user manipulates the data capture and analyzes the results, when satisfied with the collected data, presses an approval button and ingests the data into the project repository.

Deagen et al. introduce in their paper [14] an Autonomous Experimental System for polymer data analysis. In the system, there is a data ingestion module based on interfaces and data standards that allow end users with little knowledge in technology to use the platform without difficulty. The result of the project is a REST API developed in Python language for data access and a web interface to browse and organize the data visually.

Regarding data ingestion, the Palys & Palys [15] article presents the SAP system module called SAP Document and Reporting Compliance (DRC), used for electronic taxes. The module has support for data inputs such as XML, JSON and direct reporting. Throughout the process, there is transparency and traceability of the data by the user, allowing a data ingestion that is configurable and can be automatized by the customer.

B. RQ2 – What are the metrics observed for data ingestion performance analysis?

Among the analyzed papers, the only one that brought the performance metrics was the one developed by Hacker et al. [13]. The authors present that in cases where real-time data is needed, due to the requirement for data availability, synchrony with time becomes an essential factor. Based on that, ingest reliability, speed, and latency become relevant factors.

C. RQ3 – Which are the most suitable techniques for automated data ingestion in AutoML systems?

Analyzing the papers to answer the research question 3, it was considered as the main requirements for AutoML systems as to bring support to non-expert users, such as user interfaces and the possibility for easy data ingestion automation. The work of Otamendi et al. [12] provides a data analysis service focused on non-expert users. However, it is indicated in the article that the project has certain limitations regarding the automated data ingestion, due to the Open Data Cube...
used in the project, which makes the automated data ingestion module not user-friendly for non-expert users, due to the need that the user previously knows the database to configure it. This problem is mitigated through a model that performs the extraction of metadata from other databases so that inference can be made about the characteristics of the current metadata, alleviating this need for the user.

In the paper by Hacker et al. [13] several features are listed that made the prototype a success among non-expert users, such as:

- A reliable, functional, and performing data management framework for data collection and analysis;
- When a real-time data analysis is needed, time synchrony is a key factor;
- The tool should be easy to operate without the need for special expertise, allowing the user to review, check and approve data before the ingestion.

The programmatic interface presented in the Deagen et al. [14] article allows developers to configure automated data ingestion, while for end users, there is a GUI that allows visual validation of the data that has been uploaded and easy data navigation. It is important to emphasize that the project was developed with a focus on the end users (non-experts), who participated in the development stage through interviews.

Although not indicated to focus on non-expert users, the SAP system presented in the work of Palsys & Palsys [15] brings the possibility of automation of data ingestion, abstracting the difficulties of this step of the process, thus making this section of the system relevant to the current research.

V. Conclusion
As presented throughout this paper, the democratization of data analysis is a decisive factor for organizational success today, since it empowers decision makers. In this context, this paper look to contribute by defining the most suitable data ingestion techniques based in what is used in the literature to abstract this step for non-expert users in AutoML systems.

Exploring the research results, we can see that automated data ingestion can be developed in several ways, almost always focusing on the non-expert user. From the paper analysis we can conclude that important metrics for the evaluation of data ingestion are ingestion reliability, speed, and latency. Relevant features to be present in such systems can be observed as the navigation on the data to be ingested with a visual interface, providing the user the ability to approve or disapprove the data ingestion, based on the visualized data and have a wide range of data input options such as CSV, XML, JSON, etc.

In addition to the general features, important factors can be observed regarding the support of the non-expert user, the focus of AutoML systems, in the data ingestion stage. These factors can be listed as the existence of a user interface that is easy to use, the inference of metadata from the databases being used and the reliability of the data that is being manipulated.

Machine learning has an increasingly important role in society, so, the development of automated techniques for building ML models will be critical for advancing the field. As the field of AutoML continues to grow and evolve, we can expect to see many more improvements in the area of automated data ingestion.

VI. Future Work
This work acts as the initial phase of an ongoing research that aims to develop an automated data ingestion module to support business users in obtaining data for the development of activities in AutoML software.

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

REFERENCES
DGNN: Dynamic Graph Neural Networks for Anomaly Detection in Multivariate Time Series

Bowen Chen¹, Hancheng Lu¹, Yuang Chen¹, Haoyue Yuan¹, and Minghui Wang²
¹ University of Science and Technology of China, Hefei, China
² AI Institute of H3C Technologies Co., Ltd.,
{chenbowen, yuangchen21, yhyue}@mail.ustc.edu.cn, hclu@ustc.edu.cn, mhwang@h3c.com

Abstract—In recent years, there has been significant progress in the importance of anomaly detection for multivariate time series in industrial applications. However, there are still limitations. Although deep learning methods have improved anomaly detection in high-dimensional multivariate time series, they are computationally expensive and do not explicitly learn the relational structure between sequences. In this paper, we propose an unsupervised anomaly detection algorithm called Dynamic Graph Neural Networks (DGNN). Firstly, we propose a data-driven method of generating "subgraphs" to interpret interior correlations between sequences, instead of using the traditional method of a fully connected graph. Secondly, we introduce a novel Graph Attention Networks based on correlation to fuse neighbor sequence features. Experimental results on five public datasets demonstrate that our method consistently achieves state-of-the-art performance compared to other baseline methods, while reducing the edges of the graph by nearly 70%.

Index Terms—time series, anomaly detection, graph attention network, correlation coefficient

I. INTRODUCTION

With the advent of the Internet of Things (IoT) era, the recording of time-series data [1]–[3] has experienced a significant increase. This data is collected from various domains, including operations and maintenance, economics, and transportation, and encompasses metrics such as CPU usage, online shopping request numbers, resident purchasing and expenditure indexes, traffic flow, and average vehicle speed. Many time series exhibit periodic behavior, such as human heartbeat, rush-hour traffic, and tides linked to lunar and solar cycles. Anomaly detection in time series data has emerged as a critical topic in data mining, with diverse applications across industries [4]–[6]. Accurate and timely anomaly detection enables industrial systems to continuously monitor key indicators in time series data and issue early warnings for potential events. While detecting anomalies in a single time series is relatively straightforward, with the advancement and expansion of systems, an increasing number of time series data for key indicators (KPIs) are being recorded. As the dimensionality of time series data grows, manual monitoring by humans becomes increasingly challenging. Therefore, there is a need for automatic anomaly detection methods that can efficiently identify anomalies in high-dimensional data and provide explanations to human operators.

Recent techniques based on Graph Neural Networks (GNNs) have improved anomaly detection in high-dimensional datasets. For example, MTAD-GAT (Zhang et al. 2020) [7] learns the relationship between feature dimension and time dimension of multivariate time series through Graph Attention Network (GAT) (Velickovic et al. 2017) [8], and finds potential anomalies in high-dimensional multivariate time series. GDN (Deng et al. 2021) [9] employs Embedding technology to learn the potential relationship of multivariate time series and represent the relationship of each sequence time using embedding, which is convenient for human operators to diagnose and locate anomalies.

However, the existing methods use GAT to directly learn the attention coefficient and aggregate the features of the neighboring time series, which is coarse and inefficient. Because the time series in real-world scenarios have various modes, such as stationary, irregularly fluctuating, or periodic. These existing GAT-based methods all use a fully connected graph, which results in a waste of computing resources for GAT graph learning and may lead to inaccurate aggregation results. There is a wealth of information in multivariate time series, such as the possibility of similar changing trends between series, as well as slow-changing trends within individual series. However, since GAT was not designed for multivariate time series, it cannot fully exploit the information contained within them. Existing anomaly detection methods utilize GAT to fuse multivariate time series features and can only roughly extract some of their internal features.

To overcome these challenges, we propose Dynamic Graph Neural Networks (DGNN), a novel framework for detecting anomalies in multivariate time series. DGNN utilizes dynamic subgraphs that are generated based on correlations, which replace static fully connected graphs, and Adaptive Graph Attention Network(AGAT), a correlation-based alternative to traditional GAT. Compared to previous methods, dynamic subgraphs more accurately depict the proximity between each time series and help to narrow down the potential locations of anomalies. Furthermore, when GNN is employed for feature fusion, fewer edges result in less interference from independent sequences and faster convergence rates of the GNN model. AGAT leverages correlations to calculate attention coefficients and is effective in extracting features from multiple time series compared to GAT.

In summary, our main contributions are as follows:

- We propose DGNN, a dynamic graph neural network approach that enables faster and more accurate learning of
temporal relationships between time series than existing methods.

- We introduce Dynamic Subgraph Generation (DSG), a real-time approach for generating graphs between sequences based on input time series, dynamically adapting to changing data.
- We present the Adaptive Graph Attention Network (AGAT), a novel approach for effectively capturing features from multiple time series data by leveraging correlations to compute attention coefficients.
- We conduct extensive experiments on five public real-world datasets. Our results demonstrate the superior performance of DGNN over state-of-the-art baseline methods.

II. RELATED WORK

PCA (Shyu et al. 2003) [10] and IForest (Liu et al. 2008) [11] are machine learning-based methods for anomaly detection. PCA detects anomalies by reducing dimensionality and projecting data back to the original space, and comparing the deviation between original data and reconstructed data. It is fast but may lose information. IForest is an ensemble-based fast anomaly detection algorithm that determines anomalies by constructing randomly generated isolation trees, with shorter path lengths indicating a higher likelihood of anomalies. However, IForest is not suitable for high-dimensional data as randomly selecting dimensions may reduce algorithm reliability.

To address the issue of high-dimensional data loss, deep learning methods have been introduced. Time series anomaly detection methods based on deep learning, such as LSTM (Malhotra et al. 2008) [12], DAGMM (Zong et al. 2018) [13], and USAD (Audibert et al. 2020) [14], are able to incorporate contextual information of time series for prediction. Transformer-based models like TranAD (Tuli et al. 2020) [15] and Anomaly Transformer (Xu et al. 2021) [16] can better learn the contextual information of time series, leading to significant performance improvement. However, they still do not fully utilize the relationships among multiple time series. For example, Anomaly Transformer proposed an anomaly detection model based on the differences between associations, but it only focuses on the contextual information of individual time series without considering the relationships among sequences.

In recent years, GNNs have achieved success in handling graph-structured data. GNNs consider the state of neighboring nodes and utilize the latent relationships among nodes. Graph Convolutional Networks (GCNs) (Kipf and Welling 2016) [17] model node features by aggregating representations of one-step neighbors. GATs extend the approach of GCN by introducing attention functions to compute weights for different neighbors during the aggregation process. GDN (Deng et al. 2017) is a multivariate time series anomaly detection method based on GAT, but it uses a fully connected graph as the input to the GAT network, treating all time series as having the same relationship, which does not align with the actual scenario.

Using fully connected graph as input leads to difficulties in model training and slow convergence. Additionally, GAT is not specifically designed for time series, and its effectiveness in feature extraction for time series is limited.

III. FRAMEWORK OF DGNN

A. Problem Statement

We represent the network as a graph $G = (V,E)$, where $V$ is a finite set of nodes with $|V| = N$, corresponding to the observation of $N$ sensors, and $E$ is a set of edges. The observed graph signal $X_G^{(t)} \in \mathbb{R}^{N \times d}$ indicates the observation of graph information $G$ at time step $t$, where each element represents the observation of $d$ sensor features. In this paper, we utilize a forecasting-based model to detect anomalies by comparing the errors between predicted and actual values. The objective of forecasting is to learn a function $f$ from the previous $T$ observations, which can predict the next step from $N$ correlated sensors.

$$[X_G^{t−T+1}, ..., X_G^t] \xrightarrow{f} [X_G^{t+1}] \quad (1)$$

Our training dataset comprises sensor data (i.e., multivariate time series) from $N$ sensors across $T_{train}$ time steps. The sensor data is denoted as $s_{train} = [s_{train}^{(1)}, ..., s_{train}^{(T_{train})}]$ and is utilized to train our approach. At each time step $t$, the sensor values $s_{train}^{(t)} \in \mathbb{R}^N$ form an $N$-dimensional vector representing the values of our $N$ sensors. Consistent with the standard unsupervised anomaly detection formulation, the training data is assumed to consist solely of normal data.

Our goal is to detect any irregularities in the testing data, which is collected from the same $N$ sensors but covers a distinct set of $T_{test}$ time intervals. The test data is denoted by $s_{test} = [s_{test}^{(1)}, ..., s_{test}^{(T_{test})}]$. The label is a collection of $T_{test}$ binary values that indicate whether each test time interval contains an anomaly or not. Specifically, $a(t) \in \{0, 1\}$, where $a(t) = 1$ indicates that the time interval at $t$ is anomalous.

B. Overview

Our DGNN (Dynamic Graph Neural Network) method aims to cluster sensors with comparable characteristics into subgraphs and subsequently detects and clarifies discrepancies from learned patterns. The method comprises three primary components:

1) Dynamic Subgraph Generation: Builds subgraphs consisting of sensors with similar characteristics.

2) AGAT Features Learning: Uses Adaptive GAT based on correlation coefficients to learn representations from neighboring nodes.

3) Forecasting: Predicts the next step values of each sensor by leveraging a fusion representation generated by AGAT.

Figure 1 provides an overview of our framework.
C. Data Preprocessing

To make our model more robust, we apply data normalization and segment the data into time-series windows for both training and testing. We normalize the time-series data using the following equation:

$$ \tilde{x} = \frac{x - \min(X)}{\max(X) - \min(X)} $$

Here, $\max(X)$ and $\min(X)$ represent the maximum and minimum values in the training dataset, respectively.

D. Dynamic Subgraph Generation

One of the primary goals of our framework is to establish subgraphs between sensors. To achieve this, we utilize an undirected graph, where the nodes represent sensors and the edges represent dependency relationships between them. An edge connecting two sensors indicates that they can model each other’s behavior.

In our approach, we employ the correlation coefficient to establish subgraphs. Specifically, we connect two nodes with an edge if the correlation coefficient of their features exceeds a specified threshold. The correlation coefficient measures the degree of linear correlation between two sets of data variables, denoted as $X$ and $Y$. It is the ratio of the product of their covariance and their standard deviations. Essentially, it is a normalized measure of covariance, with a range of values between -1 and 1. The correlation coefficient is defined as follows:

$$ \text{Corr}(X,Y) = \frac{\text{Cov}(X,Y)}{\sqrt{\text{Var}[X]\text{Var}[Y]}} \quad (3) $$

where $\text{Cov}(X,Y)$ represents the covariance of $X$ and $Y$, and $\text{Var}[X]$ and $\text{Var}[Y]$ represent the variance of $X$ and $Y$, respectively.

$W_{i,j}$ represents the relationship between the two nodes $i$ and $j$, and $W_{i,j} \in \{0,1\}$, indicating that the two nodes have an edge ($W_{i,j} = 1$) or no edges ($W_{i,j} = 0$), respectively. It is defined as follows:

$$ W_{i,j} = \text{Corr}(x_i,x_j) > \text{threshold} \quad (4) $$

Where $\text{threshold}$ is an experience value, the default is 0.65.

Graph neural networks require the input of the structure of a graph. The lower part of Figure 2 shows a comparison between two methods. As shown in the figure, there are time series generated by seven sensors. The left side represents a model based on GAT that takes a fully connected graph as input, while the right side represents DGNN that uses a graph constructed by DSG as input. The subgraph constructed by DSG has a closer relationship than the fully connected graph.

By DSG, we split a fully connected graph into multiple subgraphs with similar features. Next, we will define our adaptive graph attention network which utilizes those subgraphs.

**Algorithm 1: Dynamic Subgraph Generation**

```
Input: N time series from $V(|V| = N)$
1 W Initialization, reset to zero matrix;
2 E Initialization, reset to empty set $E$;
3 for $i = 1,2,\ldots,n$ do
4     for $j = i + 1, i + 2,\ldots,n$ do
5         // CDC: Correlation Distance Calculation defined in Equation 3;
6         dist$_{i,j} = \text{CDC}(V_i,V_j)$; (Equation. 3)
7         if dist$_{i,j} > \text{threshold}$ then
8             $W_{i,j} = 1$; $W_{j,i} = 1$;
9         end
10     end
11 end
12 for $i = 1,2,\ldots,n$ do
13     for $j = 1,2,\ldots,n$ do
14         if $W_{i,j} = 1$ then
15             $E \leftarrow E \cup \{i,j\}$;
16         end
17 end
18 end
19 return Edge list $E$ of Subgraph $\mathcal{G}$.
```
E. AGAT Features Learning

We propose a graph attention-based feature extractor that captures the relationships between neighboring sensors by fusing a node’s information with its neighbors based on subgraphs. Unlike existing graph attention mechanisms, our feature extractor uses the correlation coefficient.

Our AGAT takes a set of node features, denoted by \( h = h_1, h_2, \ldots, h_N \), where \( h_i \in \mathbb{R}^{F} \) and \( N \) is the number of nodes with \( F \) features in each node. The AGAT produces a new set of node features, denoted by \( h' = h'_1, h'_2, \ldots, h'_N \), where \( h'_i \in \mathbb{R}^{F'} \) may have a potentially different cardinality than \( F \).

To increase the expressive power required to transform input features into higher-level features, at least one learnable linear transformation is necessary. Initially, a shared linear transformation is applied to every node, parametrized by a weight matrix, \( W \in \mathbb{R}^{F' \times F} \). Next, self-attention is performed on the nodes using the correlation coefficient, which computes attention coefficients, \( e_{i,j} \), indicating the significance of node \( j \)'s features to node \( i \).

\[
e_{i,j} = Corr(W \vec{h}_i, W \vec{h}_j) \quad (5)
\]

To incorporate graph structure into the mechanism, masked attention is used, computing \( e_{i,j} \) only for nodes \( j \in N_i \), where \( N_i \) represents the neighborhood of node \( i \) in the graph. In all experiments, the neighborhood will include only the first-order neighbors of node \( i \) (including \( i \)). To make coefficients comparable across different nodes, the softmax function is used to normalize them across all choices of \( j \) as follows:

\[
a_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k \in N_i} \exp(e_{i,k})} \quad (6)
\]

In our experiments, the attention mechanism is computed as the correlation coefficient and is then subjected to the LeakyReLU nonlinearity with a negative input slope of \( \alpha = 0.2 \). This allows us to express AGAT as follows:

\[
a_{i,j} = \frac{\exp(\text{LeakyReLU}(Corr(Wh_i, Wh_j)))}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(Corr(Wh_i, Wh_k)))} \quad (7)
\]

Once the attention coefficients have been normalized, they are used to compute a linear combination of the features corresponding to them, which serves as the final output features for each node (with the possibility of applying a nonlinearity \( \sigma \)):

\[
\vec{h}'_i = \sigma(\sum_{j \in N_i} a_{i,j} Wh'_j) \quad (8)
\]

Figure 3 illustrates the attention mechanism based on correlation used in AGAT.

F. Forecasting-based model

The forecasting model predicts the value for the next step by utilizing recurrent neural networks (RNNs) to model temporal dependency. Specifically, we employ Gated Recurrent Units (GRUs) (Chung et al., 2014), a simple yet powerful variant of RNNs. The predicted output of the forecasting model is denoted as \( \hat{s}^{(t)} \). To minimize error, we use the Mean Squared Error (MSE) loss function between the predicted output \( \hat{s}^{(t)} \) and the observed data \( s^{(t)} \):

\[
L_{MSE} = \frac{1}{T_{train}} \frac{1}{w} \sum_{t=w+1}^{T_{train}} ||\hat{s}^{(t)} - s^{(t)}||^2_2 \quad (9)
\]

Here, \( w \) represents the sliding window size of the input time series.

G. Anomaly Detection

To ensure a fair comparison, we employ the Peak Over Threshold (POT) method to automatically and dynamically select the threshold. This method is essentially a statistical approach that uses “extreme value theory” to fit the data distribution with a Generalized Pareto Distribution and determine appropriate value at risk for dynamically selecting threshold values. For each dimension \( i \), the anomaly diagnosis label \( y_i \) and detection result \( \hat{y}_i \) are defined as:

\[
y_i = 1(s_i >= \text{POT}(s_i)) \quad (10)
\]

To compute the overall anomalousness at time tick \( t \), we aggregate over sensors using the max function. We use max as anomalies may affect only a small subset of sensors or even a single sensor:

\[
y = \max(y_i) \quad (11)
\]

Thus, if any of the \( N \) dimensions are anomalous, we label the current timestamp as anomalous.

IV. EXPERIMENTS

A. Datasets

In our experiments, we use several publicly available datasets, and their characteristics are summarized in Table I. For example, the SMAP dataset has 25 dimensions, with a training dataset length of 135,183, a test dataset length of 427,616, and an anomaly ratio of 13.13%.
### TABLE I

**Dataset Statistics**

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Features</th>
<th>Train</th>
<th>Test</th>
<th>Anomalies(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSL [4]</td>
<td>55</td>
<td>58317</td>
<td>73729</td>
<td>10.72</td>
</tr>
<tr>
<td>SMD [18]</td>
<td>38</td>
<td>708405</td>
<td>708420</td>
<td>4.16</td>
</tr>
<tr>
<td>SWaT [19]</td>
<td>51</td>
<td>496800</td>
<td>449919</td>
<td>11.98</td>
</tr>
<tr>
<td>WADI [20]</td>
<td>123</td>
<td>1048571</td>
<td>172801</td>
<td>5.99</td>
</tr>
</tbody>
</table>

### TABLE II

The F1-score(%) results for anomaly detection on five publicly available datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>ASD</th>
<th>MSL</th>
<th>SMAP</th>
<th>SWaT</th>
<th>WADI</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA [2003]</td>
<td>82.28</td>
<td>91.16</td>
<td>24.67</td>
<td>58.64</td>
<td>31.86</td>
</tr>
<tr>
<td>IForest [2008]</td>
<td>75.70</td>
<td>84.56</td>
<td>67.18</td>
<td>73.50</td>
<td>69.64</td>
</tr>
<tr>
<td>LSTM-NDT [2018]</td>
<td>97.63</td>
<td>88.23</td>
<td>90.25</td>
<td>57.39</td>
<td>73.31</td>
</tr>
<tr>
<td>DAGMM [2018]</td>
<td>71.33</td>
<td>92.54</td>
<td>71.71</td>
<td>75.02</td>
<td>18.58</td>
</tr>
<tr>
<td>MTAD-GAT [2018]</td>
<td>97.83</td>
<td>97.33</td>
<td>94.77</td>
<td>77.79</td>
<td>69.75</td>
</tr>
<tr>
<td>USAAD [2018]</td>
<td>62.30</td>
<td>86.75</td>
<td>76.15</td>
<td>55.37</td>
<td>50.67</td>
</tr>
<tr>
<td>GDN [2021]</td>
<td>96.31</td>
<td>96.52</td>
<td>94.33</td>
<td>27.29</td>
<td>73.59</td>
</tr>
<tr>
<td>TranAD [2021]</td>
<td>97.17</td>
<td>96.66</td>
<td>93.61</td>
<td>58.35</td>
<td>74.01</td>
</tr>
<tr>
<td>AnomalyTransformer [2022]</td>
<td>90.87</td>
<td>95.51</td>
<td>93.45</td>
<td>55.82</td>
<td>64.37</td>
</tr>
<tr>
<td><strong>DGNN</strong></td>
<td><strong>98.21</strong></td>
<td><strong>97.94</strong></td>
<td><strong>96.34</strong></td>
<td><strong>77.80</strong></td>
<td><strong>77.19</strong></td>
</tr>
</tbody>
</table>

### B. Baseline

We conduct a comparison between our AGAT and a wide range of state-of-the-art multivariate time-series anomaly detection models, including: 1) Statistics-based models: IForest. 2) Reconstruction-based models: PCA, DAGMM, and USAAD. 3) Forecasting-based models: LSTM-NDT, MTAD-GAT, GDN, TranAD, and Anomaly Transformer.

### C. Evaluation Metrics

We use F1-Score (F1) over the test dataset and its ground truth values to evaluate the performance of our method and baseline models. The F1-Score is calculated as 
\[ F1 = \frac{2 \times Prec \times Rec}{Prec + Rec} \]
where \( Prec = \frac{TP}{TP + FP} \) and \( Rec = \frac{TP}{TP + FN} \). Here, TP, TN, FP, FN represent the numbers of true positives, true negatives, false positives, and false negatives, respectively.

### D. Results

Our experimental results, presented in Table II, show outstanding performance on all five datasets with the F1 score calculated based on POT. Our model outperforms LSTM-NDT due to its neighbor feature fusion mechanism, while MTAD-GAT falls short due to the fully connected graph causing the GAT model to extract internal connections. The importance of dynamic subgraph generation is emphasized. GDN’s poor performance on the SWaT dataset is attributed to the ineffective use of a fully connected layer as a predictive model. Figure 4 shows a significant reduction in the number of edges generated by DSG, with the MSL dataset experiencing a 98% reduction.

### E. Ablation

To investigate the necessity of each component in our method, we gradually removed them to observe how the model’s performance was affected. Firstly, we evaluated the importance of the learned graph by substituting it with a static complete graph, where every node is connected to all the other nodes. Secondly, to assess the significance of the , we disabled the attention mechanism and aggregated the information from neighboring nodes using equal weights. Lastly, to assess the significance of the Forecasting-based Model, we replaced GRU with a fully connected layer. The outcomes are summarized in Table III, revealing the following findings:

- Replacing the learned graph structure by DSG with a complete graph degrades performance in both datasets. The effect on the WADI dataset is more pronounced. This indicates that the graph structure learner enhances performance, especially for large-scale datasets.
- Removing the attention mechanism degrades the model’s performance most in our experiments. Since sensors have very different behaviors, treating all neighbors equally introduces noise and misleads the model. This verifies the importance of the graph attention mechanism.

These findings suggest that the adoption of a learned graph structure, adaptive attention mechanisms, and forecasting-based model all contribute to the accuracy of our DGNN method, which provides an explanation for its superior performance over baseline methods.
Our algorithm outperforms existing SOTA algorithms on five public datasets. In the future, we plan to apply DSG to more GNN models, replacing fully connected graphs with dynamic subgraphs to enhance their performance.

VI. ACKNOWLEDGEMENT

This work was supported by National Key R&D Program of China under Grant 2020YFA0711400, and National Science Foundation of China under Grant U21A20452, U19B2044.

REFERENCES

[6] R. Wu and E. Keogh, “Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress,” IEEE Transactions on Knowledge and Data Engineering, 2021.
**FOODIE: A Data-centric Sifting Framework for Social Media Analytics**

Dolly Agarwal*  
Ai Palette Pte Ltd  
Singapore  
dolly@aipalette.com

Abdullah Al Imran†  
Ai Palette Pte Ltd  
Singapore  
imran@aipalette.com

Kasun S Perera‡  
Ai Palette Pte Ltd  
Singapore  
kasun@aipalette.com

*ORCID: 0000-0001-8004-8650  
†ORCID: 0000-0002-3781-8178  
‡ORCID: 0009-0005-3076-1059

Abstract—There has been a great deal of research conducted in the past on utilizing social media analytics to derive consumer insights and understand their behaviors. However, when such studies are applied to real-world data in an industrial use-case, the results are often found to be incorrect and erroneous. This is a major barrier for companies that provide social media analytics-based solutions to customer-centric industries such as Food and Beverage (FnB). One of the key causes of this barrier is the failure to appropriately process and curate raw social data prior to analytics. In this study, we discuss the challenges we encountered when dealing with social data throughout our industrial experience and propose a standard solution - FOODIE. This is a framework specifically designed for the FnB industry to process social conversation data accurately and in a standard manner prior to perform various downstream tasks on it. The three stages of this paradigm are preparation, sifting, and evaluation. Through this framework, we have reduced the data to error ratio from 8.76% to 0.01% which is quite significant given the volume of the data. While this framework is designed for the FnB space, it can be customized to suit the needs of various other industries.

Index Terms— Data-centric AI, Social Media Analytics, Data Engineering, Sifting Framework, Natural Language Processing (NLP), Classification, Transfer Learning, Sentence Transformers, Clustering, Machine Learning (ML), Triplet Loss, Topic Modeling

I. INTRODUCTION

Social media analytics in food and nutrition has recently gained attention [24]. People love to document, photograph, and share their eating habits, emotions and advices about food [3]. This serves as a crucial data source to the FnB companies for monitoring food trends, opinions, eating behaviors and market potential. Numerous studies have also been conducted on the use of algorithms to extract additional insights from social media data [7], [14], [17], [23]. However, when these methods are applied in real-world applications by businesses, they fail to deliver valid results. One common stumbling block that hampers these efforts is data quality and reliability issues. Thus, ensuring the accuracy, and trustworthiness of data is currently a major problem for business. Despite the importance of assessing and governing data quality for social media analytics, there aren’t many studies that specifically address this issue [3], [20]. Moreover, the ideal way to assess the impact of poor data quality is to employ some downstream tasks, but in order to do that, we typically need a test set made up of test samples and their ground truth labels. Standard use cases meet this criteria, however many real-world instances contain test data that is not labeled, making it impossible to apply typical model evaluation techniques.

In this work, we present our industrial experience, the challenges we faced and solutions we implemented while working with social conversations to prepare it for downstream tasks like FnB trend forecasting, and innovation of potential new products. We extract social media posts from Twitter, Instagram and Facebook using food-related keywords and hashtags which is the only available way to extract social media data from such platforms. These keywords/hashtags are listed by our domain experts and research analysts. Some examples of hashtags used to extract data from USA are ‘bostonfoodies’, ‘newyorkfoodguide’, ‘miamifood’. But, this method of data extraction comes with a trade-off between high precision which means employing targeted keywords to get fewer, but more accurate results; and recall which utilizes broad terms to produce a lot of irrelevant results. In reality we have experienced that it is cheap to collect the data, but expensive to annotate and sift them [7]. We examine these problems in-depth in this study and offer ML-based and adaptive rule-based solutions. However, social interactions are challenging to interpret due to the significant use of slang, typos, concatenated words, virality, and various forms of promotional content from businesses. We look into these issues as well and offer some heuristics-based solutions.

Finally, we devise an evaluation strategy to approximate precision-recall; an ideal framework should identify relevant data while returning as close to zero false positives as possible. It is worth noting that spot checking for false positives is simple and part of quality assurance process in many businesses, but there are not many ways to systematically and statistically assess false negative rates. Hence, we propose two metrics: data-to-error ratio and relevancy to approximate data quality without much annotated data.
In summary, this paper makes the following contributions:

- We design an end-to-end framework: FOODIE to prepare social media data for downstream analytical tasks.
- We show how we can use heuristics and ML based approaches in the denoising and sifting process.
- We propose a novel method to evaluate social data quality quantitatively when it is expensive to get annotations and hence not straightforward to measure metrics like Precision-Recall on a downstream task.

The rest of the paper is organized as follows: Section II details the challenges addressed in this work. In Section III, we present our proposed framework, details of which is described from Section IV to Section V. Section VI explains the validation and evaluation aspect of our framework, and finally, in Section VII we conclude.

II. CHALLENGES ADDRESSED

Empirical investigation done in [6] reveals that a firm’s intention for big data analytics can be positively affected by its competence in maintaining the quality of data. Good quality data provides better leads, better understanding of consumers and trends. Some of the challenges with social data have been looked at in [9], [14], [17], [22]. There also has been a few attempts to provide a quality framework for social media data as in [5], [20]. To the best of our knowledge, this is the first body of work that focuses on data quality challenges of social media analytics for FnB companies and provides a framework to combat it. We detail the challenges as below:

**Homographs:** Since the data is extracted using a keyword based approach, difficulty arises when the phrases containing homographs are encountered [2]. All posts with words that are written/spelled the same but have different meanings are considered while collecting. Some of these posts may not be food-related at all. For example - ‘Nut’ can refer to both dry fruit and a type of hardware tool.

**Context variation:** Food-related items can be used in other domains especially in skin-care. For example: Green Tea is consumed as a beverage, it can also be the ingredient of a skincare product used by customers.

**Business/Promotional posts:** Many businesses use social media for advertising, promotional giveaways etc. The redundant and invalid information, such as advertising posts are not the Voice of the customer (VoC) and may alter the analytics [3], [22]. Hence these posts should be identified and handled carefully.

**Viral/Influencers posts:** Some posts have more engagement (likes, comments, shares) than others as they may be posted by influencers or popular figures on the internet. Such posts are important to handle appropriately else they can skew trend analytics [13].

**Irrelevant Hashtags:** People use trending hashtags to try to reach as many users as possible, even when their content has nothing to do with the hashtag in question i.e hashtags are often misused in an effort to get more engagement. This results in a lot of False Positives in the database when we extract social posts using the hashtag based approach. For example: ‘New music coming soon to all streaming platforms. #Music #NewArtist #Foodie #HealthyFood’.

Apart from these data challenges, another common roadblock is data quality assessment and impact evaluation. Experimenting on some downstream tasks is the optimum way to evaluate the effects of bad data quality, but in order to do so, we normally need annotated data which are expensive to curate. Since real-world scenarios involve data that is not labeled, it makes it difficult to employ conventional methods for model evaluation. Hence, we worked on a new evaluation metric to approximate the data quality (Section VI).

III. PROPOSED FRAMEWORK

Figure 1 shows our framework FOODIE that is deployed in production. The overall framework is divided into 3 phases as listed below:

**Data Preparation:** This is the first phase that helps to remove and fix noise unique to social media data. Detail about how we do it is discussed in section V.

**Sifting:** We filter out noisy and irrelevant data samples in this phase and bucket the relevant samples into food categories. We further remove duplicates to prevent over-indexing of a few samples. These steps are elaborated in section V.

**Validation:** Finally, we validate the data quality returned by the framework. We discuss this phase in detail in Section VI.

IV. DATA PREPARATION PIPELINE

Social conversations can have plenty of anomalies like typos, repeated characters in a word, spelling errors, concatenated words, short document lengths compared to the long text and normal documents. Inspired by the SMDCM model [17], we divided our cleaning process into two parts: Noise Removal and OOV handling. The aim of these data cleansing stage is to eliminate noise and transform the user posts into a formal standardized English language.

**Noise Removal:** We used regex to transform all user mention to ⟨username⟩, urls to ⟨url⟩, phone number to ⟨phone⟩. We removed all symbols, punctuations and transformed emojis/emoticons to its word representation. We also separated the text into text_only and hashtags of a post into two separate fields. If a hashtag appears within a text, we consider it as part of the text_only. For example: if a post is ‘chocolate #coffee #cake made for hubby’s birthday #foodie #instafood #instafoodie #streetfood’ then text_only field will have ‘chocolate coffee cake for hubby’s birthday’ and hashtags field will have ‘foodie, instafood instafoodie,'
Out-of-Vocabulary (OOV) Handling: Words might be OOV in social media data due to reasons like, typo or use of elongated words to express feelings or concatenated words to tackle word limitation of a social platform. We first transform elongated words into their original word by eliminating repeated letters. Then we use Peter Norvig’s edit distance based algorithm [1], to fix the spelling mistakes. Lastly, we used a trillion-word corpus [10] to split the concatenated words into their individual parts. For example, consider a post ‘I looove Baklava. The layers of flaky filo pastyr, pistachio filling and syrup has myheart’. In this process, ‘looove’ is transformed to ‘love’, ‘pastyr’ gets fixed to ‘pastry’ and then ‘myheart’ splits into ‘my heart’.

V. SIFTING PIPELINE

The purpose of this second phase is to filter out irrelevant data points and to mark those points which are true VoC related to FnB. We segregate this pipeline into two parts as described in section V-A and V-B depending on the type of approach used.

A. Heuristics and Statistics based sifting

1) Short-text removal: We considered only the text_only (without hashtags) data to infer the lower bound of text length to consider a post for analysis. Empirically we found that if a post contains \( \leq 15 \) characters, we do not get much information about any trend that can be used for analytics, hence we filtered out all the posts with less than 15 characters.

2) Flagging promotional content: Next, we identified promotional posts using heuristics like most frequent keywords used by businesses. Promotional content have some characteristics like they usually have a url, phone number, or contain one of the keywords like ‘order now’, ‘Call for orders’, ‘DM to know more’. According to frequency count, a few of the top keywords are ‘buy now’ (43,698), ‘order now’ (8,829), ‘Link in bio’ (7,849), ‘follow us’ (4,206). We also did a sanity check by looking at quotes randomly for each keyword to confirm our hypothesis. Additionally, we mark the usernames of these flagged posts and flag all other posts by those users as business posts. We flag these posts instead of removing them because they will be a useful source to analyze what local businesses or different brands are promoting in FnB.

3) Flagging viral content: Lastly, we use Outlier Detection to identify viral/influencer content. These are the posts that
receive more engagements than others but may turn out to be fad. Yet, these posts can bias the overall analytics if not handled well. To spot such posts, we transformed each engagement value (likes, shares and comments) in the range 1-100. Then, we calculate overall engagement score, ES using equation 1

\[
ES = \alpha \cdot \text{comments} + \beta \cdot \text{shares} + (1 - \alpha - \beta) \cdot \text{likes}
\] (1)

where \(\alpha = 0.3\), \(\beta = 0.5\), for Facebook and Twitter, while \(\alpha = 0.4\), \(\beta = 0\) for Instagram. We tuned these \(\alpha\) and \(\beta\) values empirically. These values account for the fact that it is easy to get likes in social platforms as compared to comments and shares. Then we marked all points with Z-score less than -3 or greater than 3 to be an outlier as it indicates that the data point is 3 standard deviation away from the mean. Similar to what we did for business posts, we flag these outliers instead of directly removing them because they are still information about a food trend that we do not want to lose.

B. Machine Learning based sifting

1) Semi-supervised approach to text data labeling: To prepare the labeled dataset for Food/Non-Food text classification, we used Sentence Transformers \[18\] with model ‘roberta-base-v2’ for feature extraction and UMAP \[16\] for dimensionality reduction. After reducing the dimension, we use HDBSCAN \[4\] to cluster the posts. The decision is based on the knowledge that Sentence Transformers performs well for tasks like semantic search and HDBSCAN can find clusters of varying densities (unlike DBSCAN), and be more robust to parameter selection. Then, we look at the top 10 words in each class based on frequency of occurrences and identify which classes are non-food. Example of two such clusters are shown in Figure 2.

2) Multi-modal approach to food/non-food classification: We performed multiple experiments of FnB-related post recognition on the curated dataset using visual information, textual information and the fusion of both. We employ RoBERTa-base using the text_only of labeled data to predict if the post is related to FnB or not. Apart from the text length concerns, we do not consider hashtags in this approach because we found that people/businesses use trending hashtags in their posts just to gain more engagement. These trending hashtags although food-related may not be relevant to the content they are posting. Finally, we investigate various methods for performing multi-modal fusion, analyzing their trade-offs in terms of classification accuracy and computational efficiency. We adopted a late fusion approach, where the final score FS is a linear combination of the scores provided by both image and text classification systems (Equation 2). This choice comes from the fact that for many posts/tweets, we may not have images. So, for such cases, the system should be able to infer from just one modality. Since, it is a classification problem with imbalanced classes, we use F1 score to evaluate our model. The dataset used and results of the trained classifier is outlined in Table I and II respectively.

\[
FS = \gamma \cdot \text{image_score} + (1 - \gamma) \cdot \text{text_score}
\] (2)

3) Food category classification: After sifting quality FnB quotes, we tagged each post into categories. We considered 5 food categories: Bakery, Confectionery, Beverage, Dairy and Snacks. Such categorization is beneficial because it enables us to comprehend how trends evolve across different categories (beverage vs Snacks, for example) as well as how some categories influence the others (bakery to confectionery). As a result, it is crucial to understand the context with regard to category rather than viewing it as simply Food. Also, we frame this problem as a multi-label classification problem since a single post can belong to multiple categories. For example, a post stating ‘Rainy Days call for chocolates And endless cups of Tea’ should be categorized into Confectionery as well as Beverage.

One way to formalize the problem is to think about how we can represent posts in a manner that preserves good category properties, meaning that similar items should have similar representations. A natural representation in this case would be to use embeddings that preserve intuitive relationships

\[
\text{sentence}_i = \text{sentence}_{i-1} \cdot \text{word}_i
\]

where \text{sentence}_i denotes the embedding of the sentence after adding the word \text{word}_i. This transformation is used to model the sequential nature of the text. To estimate the conditional probability of a word given the previous word, we use a language model. Given a sentence \text{sentence}_i, we want to predict the next word \text{word}_i. The probability of \text{word}_i given \text{sentence}_i is computed as

\[
\text{P}(\text{word}_i | \text{sentence}_i) = \frac{1}{\sum_{j \in \text{words}} \text{P}(\text{word}_j | \text{sentence}_i)}
\]

for each word \text{words} in the vocabulary. The language model is trained using the dataset.

\[
\text{sentence}_i = \sum_{j \in \text{words}} \text{P}(\text{word}_j | \text{sentence}_{i-1})
\]

where \text{sentence}_i denotes the embedding of the sentence. This transformation is used to model the sequential nature of the text. To estimate the conditional probability of a word given the previous word, we use a language model. Given a sentence \text{sentence}_i, we want to predict the next word \text{word}_i. The probability of \text{word}_i given \text{sentence}_i is computed as

\[
\text{P}(\text{word}_i | \text{sentence}_i) = \frac{1}{\sum_{j \in \text{words}} \text{P}(\text{word}_j | \text{sentence}_i)}
\]

for each word \text{words} in the vocabulary. The language model is trained using the dataset.

\[
\text{sentence}_i = \sum_{j \in \text{words}} \text{P}(\text{word}_j | \text{sentence}_{i-1})
\]

where \text{sentence}_i denotes the embedding of the sentence.
between items. For example, we would expect that post that talks about ‘Mango Cake’ and ‘Nutella Waffle’ to be more similar to each other than to post that talks about ‘Almond milkshake’ because the former are both Bakery food items while Milkshake is a Beverage. It has become normal practice in NLP to optimize a large pre-trained model like BERT for a downstream task. The quality of the learnt metric, however, is highly influenced by the quality of the annotated training set that we do not have access to. For uncommon classes with little data samples, this issue is more severe. We get over this problem by automatically creating samples from unlabeled data and learning a representation for the label using a Siamese Neural Network architecture with Triplet loss [12]. We hand-curated small samples for each category and then generated triplets (anchor, positive and negative) from it. We used this dataset to fine-tune Sentence Bert with model ‘all-MiniLM-L6-v2’ using triplet loss. The triplet loss seeks to push similar instances together and separate dissimilar examples in the latent space. As a result, distances between comparable samples are preserved when the encoders for the quotes are embedded into the same latent space. For the purpose of comparing the effectiveness of the triplet loss method, we experimented with 2 approaches: 1. Baseline using pre-trained Sentence Bert embedding 2. Triplet-loss based fine-tuned Sentence Bert embedding. We fine-tuned with batch size of 8 and Learning rate as 2e-5. We use the fine-tuned embeddings for multi-label text classification. Table III shows a significant improvement in multi-label classification by using triplet-loss fine-tuned SBert embedding. The category distribution samples are highly skewed with Bakery having the highest number of samples. However, using triplet loss based approach, we found that there has been noteworthy jump even for minority classes that helped spike the overall performance. This supports our approach of fine-tuning SBERT using Triplet loss by using a small hand-curated dataset.

C. Deduplication

While working with social data, we can find some posts that have exact/near duplicate content. To reduce the over-indexing on a few messages, we use a deduplication approach to apply a max cap on them. Most of the time such posts were from businesses trying to promote their product/event. Other times users just append emojis or make a small change to the text (Twitter Retweet, for example), so, doing an exact regex match doesn’t solve the problem completely [14]. We extracted the embeddings of text_only part of each post using Sentence Transformer model ‘nli-roberta-base-v2’ [19] and then computed cosine similarities between embeddings. Lastly, we consider the posts with \( \theta \leq 0.9 \) cosine similarities as duplicates.

\[
\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\| \mathbf{A} \| \| \mathbf{B} \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (3)
\]

VI. Inspection and Evaluation

We got our expert Quality Check team to spot check the data points returned by our framework. This human-validated data is used to compute the data-to-error ratio as expressed in eq. (4)

\[
\text{Data – to – Error} = \frac{\# \text{samples marked as irrelevant}}{\text{Total samples validated}} \quad (4)
\]

The data to error ratio decreased drastically from 8.76% to 0.01% after the framework deployment. However, it is simple to determine if the social media postings that have been returned are inaccurate, but it is more challenging to determine how many posts that should have been returned but were
instead overlooked by the algorithm. So, we used relevancy as defined in eq. 5 as a metric check on how much data was accepted during the sifting process.

\[
\text{relevancy} = \frac{\#\text{samples\_marked\_relevant\_by\_framework}}{\text{Total\_samples\_collected}}
\]

We consider these two metrics together as an indicator of the quality of our collection and the sifting process as shown in Figure 3. The four Taxonomies that we used to describe data quality is defined as below:

- **Insufficient**: Most data points are valid but the number of data points returned are low. This can arise if framework has high False Negatives rate to have high precision i.e high Precision, low Recall
- **Appropriate**: Sufficient and valid data points that gives good estimates on various analytics, i.e. high Precision and high recall
- **Corrupt**: When we have only a few data points and that is mostly unrelated to the study, this prohibits performing analytics or gives erratic results. This can arise if the framework returns only a few data points and those are mostly False Positives
- **Noisy**: This is the current state of the data, where the collected data has some invalid data points i.e False positives

So, the goal of our framework is to transform Noisy data into Appropriate Data. We could achieve a low data-to-error ratio (0.01%) with a relevancy score of 68.87%. Although it might seem a low relevancy score, but since we used many broad terms for data collection to get more data points, it is quite good. Thus, both metrics indicates strong evidence towards the appropriateness of data for analytics.

VII. CONCLUSION

Social media analytics have gained popularity in recent years, however many of the systems still can’t deliver accurate data for practical applications. In this work, we’ve discussed the difficulties we faced when analyzing social data for FnB analytics. We’ve provided a framework: FOODIE for cleaning and preparing the data for analytics. Moreover, with easily an-alytics or gives erratic results. This can arise if framework returns only a few data points and those are mostly False Positives.

So, the goal of our framework is to transform Noisy data into Appropriate Data. We could achieve a low data-to-error ratio (0.01%) with a relevancy score of 68.87%. Although it might seem a low relevancy score, but since we used many broad terms for data collection to get more data points, it is quite good. Thus, both metrics indicates strong evidence towards the appropriateness of data for analytics.

REFERENCES


Towards Extreme Multi-label Text Classification Through Group-wise Label Ranking

Jie Xiong*, Huiyuan Li*, Qiylan Duan†, Qihan Du*

*Renmin University of China, Beijing, 100872, China, {xiongjiezk, hyli18, duqihan}@ruc.edu.cn
†Zhihu Inc., {duanqiyuan}@zhihu.com

Abstract—Extreme Multi-label text Classification (XMC) aims to find the most relevant labels (i.e., the positives) for a document from an extremely large label set. The remaining labels are regarded as the negatives. Recently, the deep learning-based methods have been widely used to solve XMC, most of which use sigmoid as the activation function of output layer and use binary cross entropy loss as the learning objective. However, the existing methods suffer from the following limitations. First, the score of each label is predicted independently, where the label rank-missing problem is ignored. Second, the cardinalities of the positives and the negatives are extremely unbalanced in XMC, which makes the classifier more biased towards the majority one. In this paper, we use label group to denote the positive and the negative labels and propose a novel XMC model leveraging Group-wise label Ranking (X-GRank) to address those limitations. Specifically, X-GRank uses the newly proposed GRank loss to rank the label groups. Then, X-GRank solves the label imbalance problem by constraining the backward gradient amplitude between label groups. Extensive experiments show that X-GRank outperforms the state-of-the-art methods on five widely used datasets.

Index Terms—Extreme Multi-label Text Classification, Deep Learning, Group-wise Ranking

I. INTRODUCTION

Text classification is a key task in the field of natural language processing (NLP), which can be divided into multi-class and multi-label classification problems based on whether the classes/labels are mutually exclusive. Extreme Multi-label text Classification (XMC) aims to tag a document with relevant labels from an extremely large label set. XMC becomes increasingly important due to the fast-growing of online contents and the urgent need for the better organization of the messy big data. For example, Wikipedia needs to tag a new article from more than one million labels. Compared with the traditional multi-label classification methods [1], XMC focuses on how to extract uncertain positive labels from numerous candidate labels, which leaves researchers with open challenges.

In this paper, we introduce a concept of label group and view text classification from a novel unified perspective. For a given document, its associated labels are regarded as a positive group, denoted as $G_p$. Other irrelevant labels are treated as a corresponding negative group, denoted as $G_n$. $G_p$ and $G_n$ satisfy $|G_p| + |G_n| = L$, where $|G_p|$ and $|G_n|$ are the cardinalities of $G_p$ and $G_n$, and $L$ is the number of labels of the corpus. In this perspective, multi-class classification means that there is a constraint of $|G_p| = 1$, and multi-label classification allows $|G_p| >= 1$. For XMC, there may be an $L$ as large as millions, and $|G_p|$ is far less than $|G_n|$ and $L$.

Nowadays, the superiority of deep learning-based text classification methods has been frequently reported [1]–[5]. Most of the existing models stack a full connection layer on the text encoder to output the scores of each label. For multi-class classification, as shown in Fig. 1 (a), softmax and cross entropy (CE) loss always act as the activation function of the output layer and the learning objective. The scores of softmax can be viewed as a probability distribution due to the sum of all scores is 1.0, which results in maximizing the positive labels is equivalent to minimizing the negative labels and vice versa. For multi-label classification problem, to adapt $|G_p| >= 1$, softmax and CE is replaced by sigmoid and binary cross entropy (BCE) [1], [3], so that the model can predict all labels independently, as shown in Fig. 1 (b). Sigmoid and BCE loss is the currently most widely used combination in multi-label classification [1], [2]. However, this combination brings the following problems in XMC:

- The ranking learning between labels is missing, but we hope all scores in $G_p$ are greater than those in $G_n$.
- The sparse positive labels lead to serious label imbalance in XMC, which may hinder the model convergence.

To address the problems above, we explore the label ranking learning mechanism and label balancing mechanism for XMC. Specifically, we use $g_p$ and $g_n$ denote the scores of labels in $G_p$ and $G_n$, respectively. First, from the perspective of label ranking, it needs to compare the scores between $g_p$ and $g_n$, and then optimize $Obj_{raw} : g_p > g_n, i = 1, \cdots, |G_p|, j = 1, \cdots, |G_n|$. However, $Obj_{raw}$ can be hardly used as a loss function due to the non-differentiability and slow convergence. In this paper, we made a series of smooth transformations on

DOI reference number: 10.18293/SEKE23-028
and proposed a new efficient differentiable Group-wise label Ranking (GRank) loss function, which offers the comparison between label groups with linear time complexity. Subsequently, the problem of label imbalance can be addressed by GRank loss through the equivalent backward gradients between groups involved.

In addition, due to the huge label set of XMC, it is an inevitable challenge for solving the XMC with the limited computational resources. To reduce the calculation cost, dividing the training process into multiple cascaded stages, usually including a retrieving stage and a ranking stage, has become a prevalent method [2]–[4], [6]. The retrieving stage gathers a shortlist of candidate labels from the original large label set, and the ranking stage boosts the positive labels from retrieved labels. However, most of the existing methods train those stages not in an end-to-end way, such as AttentionXML [2] and X-Transformer [4], which brings unavoidable cascaded errors. In this paper, we designed a new XMC model with GRank loss, named X-GRank, which includes a retrieving task and a ranking task. We trained X-GRank in end-to-end way to eliminate the cascaded errors. Moreover, we used label features to enhance the label representation, which further improved the performance of the X-GRank. We summarize three major contributions of this paper as follows:

- We proposed a novel Group-wise label Ranking loss based on the perspective of positive and negative label groups. GRank loss addressed the rank-missing and label imbalance problems of the existing XMC models.
- We proposed a new XMC model with GRank loss. We trained X-GRank end-to-end to eliminate the cascaded errors in existing methods.
- We conducted extensive experiments on five widely datasets of XMC. The experimental results show our model achieves remarkable results on those datasets.

II. RELATED WORK

We summarized the existing work into traditional machine learning methods and deep learning methods.

The traditional methods can be divided into three classes: one-vs-all, tree-based, and embedding-based methods. One-vs-all methods, such as DiSMEC [7], have good classification results, but with heavy computational overhead. Tree-based methods generate a tree hierarchy through recursive partitioning and the representative models, including Parabel [8] and BonSai [9]. However, the cascaded errors of tree-based methods cannot be ignored. Embedding-based methods attempt to address the data sparsity issue through matrix compression and decompression, which is unfriendly to tail labels [1].

Deep learning-based methods show their strong learning ability in XMC. Many novel network structures have been proposed, including dynamic pooling technology in XML-CNN [1], attention mechanism in AttentionXML [2], negative sampling strategy in LightXML [3], and label structure learning [10]. Those methods continuously refresh the leaderboard of XMC.


In addition, the problem of tail labels is a great challenge in XMC. Tail labels are referred to the infrequently occurred labels, which have limited training samples and are harder to predict than the frequently occurred ones (referred to as head labels). Many existing XMC approaches [1], [7], [9] treat all labels with equal importance, which may lead XMC to favor head labels. Propensity score [11] is proposed to measure the model performance on tail labels. Label features are important to label learning. Effective utilization of label features in X-Transformer [4], ECLARE [5] significantly improves precision on accessible benchmark datasets.

III. PROPOSED METHOD

A. Research formulation

Given a document corpus, \( D = \{(d_i : G^p_i, G^n_i)\}_{i=1}^N \), where \( d_i \) is the \( i \)-th document, \( N \) is document size of corpus. \( G^p_i, G^n_i \) are positive and negative label group of \( d_i \). Our goal is to learn a function \( f(d_i) \in R^L \), which outputs the correlation scores of all labels for the given document. The function \( f \) should be optimized to make the scores of labels in \( G^p_i \) to be higher than that in \( G^n_i \).

B. The structure of X-GRank

We proposed a novel XMC model, named X-GRank. It consists of three components, as shown in Fig. 2: 1) a text encoder, 2) a label retriever, 3) a label ranker.

Text encoder aims to get an appropriate vector to represent a document. In this paper, we use pre-trained transformer models such as bert-base-uncased [12] as the text encoder. To make full utilize of implicit representations between each transformer layer, we concatenate the hidden states of “[CLS]” token of last five layers as the document representation, which is the same as LightXML [3].

Both label retriever and label ranker use the newly proposed loss function, GRank. So, we will first introduce the rationales of GRank, and then explain the application of GRank.

C. GRank: Group-wise ranking loss

1) Design of GRank loss: Generally, XMC uses the scored top-k labels as the prediction result. Consequently, we hope
all positive labels have higher scores than the negative labels, which can be formalized as a loss function:

$$\text{Loss} = \sum_{i=1}^{G_p} \sum_{j=1}^{G_n} \max(g^*_n - g^*_p, 0)$$  \hspace{1cm} (1)$$

However, Equation (1) has three disadvantages: 1) It is not globally differentiable, resulting in slow convergence. 2) It is not robust during training, where all label pairs are optimized independently, lacking mutual constraint of gradient amplitudes between them. 3) It has a high computational complexity, which is $O(|G_p| \times |G_n|)$. In particular, XMC has numerous labels, which will produce a huge number of label pairs.

In order to solve the above problems, we note that Equation (1) is equivalent to making the smallest positive label greater than the largest negative label, namely, $\min(g^*_p) > \max(g^*_n)$. Therefore, the reformed loss function is:

$$\text{Loss} = \max(0, \max(g^*_n) + \max(-g^*_p))$$  \hspace{1cm} (2)$$

Due to Equation (2) only includes the $\max$ function and is not differentiable, we use the softened smooth maximum function $\log$-sum-$\exp$ (LSE) [13] to approximate the $\max$ function. LSE is defined as:

$$LSE(x_1, \ldots, x_n) = \log(\exp(x_1) + \cdots + \exp(x_n))$$  \hspace{1cm} (3)$$

In particular, $\max(0, x)$ can be approximated as $LSE(0, x) = \log(1 + \exp(x))$. Therefore, we propose the following smoothing loss based on Equation (2):

$$\text{Loss} = \log(1 + \sum_{j=1}^{G_n} \exp(g^*_n)) \sum_{i=1}^{G_p} \exp(-g^*_p)$$  \hspace{1cm} (4)$$

Equation (4) is essentially designed to optimize the ranking relationship between positive and negative label groups, so we call it Group-wise Ranking (GRank) loss.

2) Analysis on GRank loss: GRank loss can effectively solve the disadvantages mentioned in Equation (1). First, GRank loss is differentiable, which uses $LSE(x)$ approximates $\max(x)$ with the limited bounds:

$$\max(x) \leq LSE(x) \leq \max(x) + \log(n)$$  \hspace{1cm} (5)$$

where $n$ is the element size of $x$, and the first inequality is strict unless $n = 1$ and the second inequality is strict unless all arguments are equal.

Second, GRank loss has solved the label imbalance problem of XMC. GRank loss is robust in training and has a balanced optimization for positive and negative groups. We denote $s = \sum_{j=1}^{G_n} \exp(g^*_n) \sum_{i=1}^{G_p} \exp(-g^*_p)$, and the derivative of Equation (4) is:

$$\frac{\partial \text{Loss}}{\partial g^*_n} = \frac{s}{(1 + s)} \sum_{i=1}^{G_p} \frac{\partial \text{Loss}}{\partial g^*_p} = \frac{-s}{(1 + s)}$$  \hspace{1cm} (6)$$

Obviously, there is a mutual constraint gradient between the two groups, which solves the label imbalance problem and can improve the robustness of the model training.

Third, GRank loss has low computational complexity, which is line time complexity with $O(L)$.

D. The retrieving stage

In this stage, we aim to obtain a shortlist of candidate labels from the original labels to reduce the computational cost in the ranking stage. A common method is to divide the original labels into some clusters, and each label belongs to one cluster. Clusters with positive labels are viewed as positive clusters, and clusters without positive labels are negative clusters. So, the cluster prediction is a small-scale XMC task, as shown in Fig. 2. The labels in the positive clusters will generate the shortlist of candidate labels.

1) Label clustering: To handle large-scale label sets, we use a Probabilistic Label Tree (PLT) to partition labels, which is widely used in LightXML [3], Bonsai [9] and AttentionXML [2]. PLT is constructed through a top-down hierarchical clustering until the termination condition is met. Each original label is the leaf node of PLT, and the parent node of the leaf node is the label clustering we need. Since a deeper PLT is more prone to error propagation, we use a two-layer PLT with the same building ways in LightXML and AttentionXML.

2) Label retrieving task: The key part of label retrieving task is to predict the positive clusters, which can be formalized as: $D = \{(d_i : G^p_{cp} \cap G^n_{cn})\}_{i=1}^{N}$, where $G^p_{cp}$, $G^n_{cn}$ are the positive and the negative clusters, respectively. We denote $g^p_{cp}$, $g^n_{cn}$ as the scores of $G^p_{cp}$ and $G^n_{cn}$. So, the GRank loss of retrieving task can be formalized as:

$$\text{Loss}_{\text{retr}} = \log(1 + \sum_{j=1}^{G_p_{cp}} \sum_{i=1}^{G_p_{cn}} \exp(-g^*_{cp}))$$  \hspace{1cm} (7)$$

E. The ranking stage

In this stage, we need to learn the label representation first. Label representation is an appropriate vector in a hyperspace to represent a label, which includes two parts in this paper. First, considering that a label may have textual features, such as title and description, we utilize a text embedding block to embed and average those text features through bag-of-words representation. Then, we add an identity embedding layer for each label, which takes the input of the unique identity number of a label in the label set. The label identity embedding is initialized randomly, and then concatenated with label text embedding as the final label representation. The overall structure of label representation learning is shown in right part of Fig. 2.

The label ranking task is to predict the scores of retrieved labels. We divide the retrieved labels into positive and negative groups, denoted as $G^*_p$ and $G^*_n$, respectively. The label scores, $g^*_p$ and $g^*_n$, are performed by the dot product of the document and label representations. We apply GRank loss again in ranking task, namely:

$$\text{Loss}_{\text{rank}} = \log(1 + \sum_{j=1}^{G^*_n} \sum_{i=1}^{G^*_p} \exp(g^*_p) \exp(-g^*_n))$$  \hspace{1cm} (8)$$

F. Model training

The retrieving task produces the candidate labels. The ranking task extracts the positive label group from the candidate
labels. Therefore, those two tasks have cascading dependencies. In order to reduce error propagation between tasks, we trained X-GRank through end-to-end learning. The intensity of model learning for those two tasks is different during model training. At the beginning of training, X-GRank should pay more attention to the retrieving task, and the ranking task should be prominent at the end of training. As a result, We designed a dynamic task weighting mechanism with linear schedule, which can be formulated as:

$$\text{Loss} = (1 - \lambda) \times \text{Loss}_{\text{retr}} + \lambda \times \text{Loss}_{\text{rank}}$$  \hspace{1cm} (9)

where $\lambda = 0.5 + e / E$, $e \in 1, \ldots, E$, $e$ is the number of current epochs, $E$ is the number of total epochs.

\section*{IV. Experiments}

\subsection*{A. Datasets and experiment setup}

We use five widely used XMC datasets\(^1\), including Eurlex-4K, AmazonCat-13K, Wiki10-31K, Wiki-500K and Amazon-670K. Those datasets follow the same processing as X-Transformer\(^2\) [4] and AttentionXML\(^3\) [2]. The detailed summary statistics of each dataset are shown in Table I, where $N_{\text{trn}}, N_{\text{tst}}$ are the numbers of training and test samples, respectively. $L$ is the number of labels, $\bar{L}$ is the average number of labels per sample, $\bar{L}$ is the average number of samples per label, $L_p$ is the percentage of tail labels (sample number per label $< 5$) in all labels, and $W_{\text{trn}}$ and $W_{\text{tst}}$ are the average number of words per-training and test sample respectively. $L_f$ denotes whether the dataset has label features.

\begin{table}[h]
\centering
\caption{Detailed Statistics of Datasets.}
\begin{tabular}{l|cc|cc|cc|cc|c}
\hline
Datasets & $N_{\text{trn}}$ & $N_{\text{tst}}$ & $L$ & $\bar{L}$ & $L_p$ & $W_{\text{trn}}$ & $W_{\text{tst}}$ & $L_f$ \\
\hline
Eurlex-4K & 15,449 & 3,865 & 3,956 & 5.30 & 20.79 & 46.07 & 1248.58 & 1230.40 & Yes \\
AmazonCat-13K & 1,186,239 & 306,782 & 13,330 & 5.04 & 438.57 & 17.95 & 246.61 & 245.98 & Yes \\
Wiki10-31K & 19,136 & 6,016 & 30,936 & 15.64 & 8.92 & 75.64 & 344.30 & 342.43 & Yes \\
Wiki-500K & 1,719,881 & 569,421 & 501,079 & 4.75 & 16.86 & 44.42 & 808.66 & 808.56 & Yes \\
\hline
\end{tabular}
\end{table}

For each dataset, we use raw texts as input, which are truncated or padded to the maximum length. For datasets with small-scale labels like Eurlex-4K, AmazonCat-13K and Wiki10-31K, the number of label cluster is set to 1 and all labels are participated in ranking. In order to make a fair comparison with LightXML, we adopt the same model integration strategy as LightXML, that is, we use three different pre-trained transformer models, including bert-base-uncased [12] roberta-base [14] and xlnet-base-cased [15], to train three sub-models to integrate. For large-scale datasets like Wiki-500K and Amazon-670K, three different label clusters are used for model training. Our model is implemented in PyTorch DistributedDataParallel\(^4\) module with NCCL as the backend.

Our Experiments are conducted on eight NVIDIA Tesla V100 GPUs (32 GB memory each). X-GRank is trained by AdamW [16] optimizer. The initial learning rate is $1E - 4$ with a warm-up proportion as 0.1. Other hyperparameters are given in Table II, where $E$ is the number of epochs, $B$ is the batch size, $d$ is the dimension of label embedding, $C$ is cluster size which is the number of labels in a cluster, $K_c$ is the top-k retrieved cluster, $L_t$ is the maximum length of input tokens of transformer model.

\begin{table}[h]
\centering
\caption{Hyperparameters of all datasets.}
\begin{tabular}{l|cc|cc|cc|c}
\hline
Datasets & $E$ & $B$ & $d$ & $C$ & $K_c$ & $L_t$ \\
\hline
Eurlex-4K & 50 & 32 & 512 & 1 & 3,956 & 512 \\
AmazonCat-13K & 20 & 32 & 512 & 1 & 13,330 & 512 \\
Wiki10-31K & 50 & 32 & 512 & 1 & 30,938 & 512 \\
Wiki-500K & 20 & 128 & 128 & 60 & 32 & 128 \\
Amazon-670K & 50 & 64 & 128 & 80 & 32 & 256 \\
\hline
\end{tabular}
\end{table}

\subsection*{B. Metrics}

We employ the widely used Recall ($R@k$) and Precision ($P@k$) \([1], [3], [4]\) as the evaluation metrics in retrieving and ranking stages, respectively. In addition, Propensity Scored Precision ($PSP@k$) \([11]\) is used to examine the performance on tail labels. $R@k$, $P@k$ and $PSP@k$ are defined as:

$$R@k = \frac{\sum_{j \in \text{rank}_k(y)} y_j}{\sum_j y_j}$$

$$P@k = \frac{1}{k} \sum_{j \in \text{rank}_k(y)} y_j$$

$$PSP@k = \frac{1}{k} \sum_{j \in \text{rank}_k(y)} y_j$$

where $y_j \in \{0, 1\}^L$ is the true binary vector, $\text{rank}_k(y)$ is the indices of the $k$, and $p_j$ is the propensity score of label $j$.

\subsection*{C. Experimental results}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Fig3.png}
\caption{Changes of recall results with varying $k$ values, where $k$ is the retrieved label size.}
\end{figure}

\begin{enumerate}
\item \textit{Results of the retrieving stage:} In retrieving stage, we evaluate the label recall results on Wiki-500K and Amazon-670K. The comparison methods include LightXML, Parabel [8] and FastXML [17]. LightXML is one of the best deep learning-based method. Parabel and FastXML are classic tree-based methods. As shown in Fig. 3, the recall of X-GRank substantially outperformed Parabel and FastXML. For example, on the Wiki-500K dataset, $R@512$ of Parabel is 59.11\%, while $R@512$ of X-GRank achieves 92.18\%, with an increase of 33.07\%. Although LightXML performed better than Parabel and FastXML, X-GRank outperformed LightXML when the retrieved label size $k$ is greater than 64 on the Wiki-500K and Amazon-670K.
\end{enumerate}

\begin{itemize}
\item [2] https://github.com/OctoberChang/X-Transformer
\item [3] https://github.com/yourh/AttentionXML
\end{itemize}
2) Results of the ranking stage: We compared X-GRank with those state-of-the-art traditional and deep learning-based methods of XMC, including DiSMEC [7], Parabel [8], Bon sai [9], XT [18], XMC-CNN [1], AttentionXML [2], X-Transformer [4], and LightXML [3]. Table III shows the results of $P@k$ on the five datasets. It is worth noting that the result of X-Transformer on Amazon-670K is not reported because of the hardware limits. In order to compare with baselines under the same conditions, we followed the metrics used in baselines to focus on top prediction by varying $k$ at 1, 3, and 5.

Compared with the traditional methods, the deep learning methods have made great progress on XMC. X-GRank, X-Transformer [4] and LightXML [3] are transformer-based approaches using a pre-trained transformer model. LightXML is the current state-of-the-art method of XMC on those five datasets. X-GRank outperforms LightXML on all datasets except the metrics $P@3$ and $P@5$ on Wiki10-31K, which demonstrate the effectiveness of our X-GRank.

D. Ablation analysis

We conducted the following ablation analysis: 1) the comparisons between GRank loss and BCE loss; 2) the contributions of label text features. All those experiments were conducted on the single model, using “bert-base-uncased” to initialize the parameters of the text encoder. We conducted the ablation study on Eurolex-4K and Wiki10-31K datasets. The results are shown in Table IV.

- X-BCE-1 w/o LF uses the combination of sigmoid and BCE loss without label text features.
- X-GRank-1 w/o LF uses GRank loss without label text features.
- X-GRank-1 w/ LF uses GRank loss and label text features.

### Table IV

<p>| Results of the ablation study of X-GRANK |
|-----------------|-----------------|-----------------|-----------------|-----------------|</p>
<table>
<thead>
<tr>
<th><strong>Datasets</strong></th>
<th><strong>Metrics</strong></th>
<th><strong>X-BCE-1 w/o LF</strong></th>
<th><strong>X-GRank-1 w/o LF</strong></th>
<th><strong>X-GRank-1 w/ LF</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Eurolex-4K</td>
<td>$P@1$</td>
<td>74.27</td>
<td>76.30</td>
<td>80.59</td>
</tr>
<tr>
<td></td>
<td>$P@3$</td>
<td>71.81</td>
<td>73.82</td>
<td>74.93</td>
</tr>
<tr>
<td></td>
<td>$P@5$</td>
<td>59.59</td>
<td>61.99</td>
<td>62.11</td>
</tr>
<tr>
<td>Wiki10-31K</td>
<td>$P@1$</td>
<td>87.71</td>
<td>88.63</td>
<td>89.63</td>
</tr>
<tr>
<td></td>
<td>$P@3$</td>
<td>76.99</td>
<td>77.72</td>
<td>77.89</td>
</tr>
<tr>
<td></td>
<td>$P@5$</td>
<td>67.63</td>
<td>68.75</td>
<td>68.10</td>
</tr>
</tbody>
</table>

### Table V

<p>| Comparison of X-GRank against the baselines on $PSP@k$ |
|-----------------|-----------------|-----------------|-----------------|-----------------|</p>
<table>
<thead>
<tr>
<th><strong>Datasets</strong></th>
<th><strong>Metrics</strong></th>
<th><strong>DiSMEC</strong></th>
<th><strong>Parabel</strong></th>
<th><strong>XMC-CNN</strong></th>
<th><strong>AttentionXML</strong></th>
<th><strong>X-GRank</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Eurolex-4K</td>
<td>$P@1$</td>
<td>38.45</td>
<td>37.20</td>
<td>37.33</td>
<td>32.41</td>
<td>44.97</td>
</tr>
<tr>
<td></td>
<td>$P@3$</td>
<td>46.20</td>
<td>44.74</td>
<td>45.40</td>
<td>36.95</td>
<td>51.91</td>
</tr>
<tr>
<td></td>
<td>$P@5$</td>
<td>50.25</td>
<td>49.17</td>
<td>49.92</td>
<td>39.45</td>
<td>54.86</td>
</tr>
<tr>
<td>AmazonCat-13K</td>
<td>$P@1$</td>
<td>51.41</td>
<td>50.92</td>
<td>51.30</td>
<td>52.42</td>
<td>53.76</td>
</tr>
<tr>
<td></td>
<td>$P@3$</td>
<td>61.02</td>
<td>54.00</td>
<td>54.60</td>
<td>56.35</td>
<td>55.63</td>
</tr>
<tr>
<td></td>
<td>$P@5$</td>
<td>65.86</td>
<td>64.00</td>
<td>64.40</td>
<td>62.83</td>
<td>62.79</td>
</tr>
<tr>
<td>Wiki-500K</td>
<td>$P@1$</td>
<td>74.50</td>
<td>73.27</td>
<td>74.12</td>
<td>74.76</td>
<td>73.38</td>
</tr>
<tr>
<td></td>
<td>$P@3$</td>
<td>79.10</td>
<td>77.48</td>
<td>78.52</td>
<td>78.13</td>
<td>75.23</td>
</tr>
<tr>
<td></td>
<td>$P@5$</td>
<td>82.90</td>
<td>81.73</td>
<td>82.49</td>
<td>81.97</td>
<td>77.58</td>
</tr>
<tr>
<td>Amazon-670K</td>
<td>$P@1$</td>
<td>26.26</td>
<td>26.36</td>
<td>27.08</td>
<td>17.43</td>
<td>30.29</td>
</tr>
<tr>
<td></td>
<td>$P@3$</td>
<td>30.14</td>
<td>29.95</td>
<td>30.79</td>
<td>21.66</td>
<td>33.85</td>
</tr>
<tr>
<td></td>
<td>$P@5$</td>
<td>33.89</td>
<td>33.17</td>
<td>34.11</td>
<td>24.42</td>
<td>37.13</td>
</tr>
</tbody>
</table>

Note: The results of all baselines are cited from Table III in LightXML [3].

### Table III

Comparing X-GRank against the state-of-the-art XMC methods on $P@k$.

- **Eurlex-4K**: $P@1$, 74.27; $P@3$, 71.81; $P@5$, 59.59
- **AmazonCat-13K**: $P@1$, 93.81; $P@3$, 79.08; $P@5$, 64.06
- **Wiki10-31K**: $P@1$, 84.13; $P@3$, 74.72; $P@5$, 65.94
- **Wiki-500K**: $P@1$, 70.21; $P@3$, 50.57; $P@5$, 39.68
- **Amazon-670K**: $P@1$, 44.78; $P@3$, 39.72; $P@5$, 36.17

Note: The results of all baselines are cited from Table I in LightXML [3].

### Table V

Comparing X-GRank against the baselines on $PSP@k$.

- **Eurlex-4K**: $P@1$, 38.45; $P@3$, 46.20; $P@5$, 50.25
- **AmazonCat-13K**: $P@1$, 51.41; $P@3$, 61.02; $P@5$, 65.86
- **Wiki-500K**: $P@1$, 74.50; $P@3$, 79.10; $P@5$, 82.90
- **Amazon-670K**: $P@1$, 26.26; $P@3$, 30.14; $P@5$, 33.89

Note: The results of all baselines are cited from Table 6 in AttentionXML [2].

1) Effects of the GRank loss: X-BCE-1 w/o LF and X-GRank-1 w/o LF use the same input. Because of the contribution of ranking between labels and the avoidance of the imbalance between positive and negative label groups, GRank loss is superior to BCE loss in multi-label classification. The ablation results have strongly verified this conclusion. As shown in Table IV, $P@1$, $P@3$, and $P@5$ of X-GRank-1 w/o LF increased by 2.03%, 2.01% and 2.40% on Eurlex-4K than that of X-BCE-1 w/o LF.

2) Effects of the label text features: Label text features are a piece of text that is potentially associated with the content of the document. Specifically, we observed that label text features are always mentioned in document on Eurolex-4K and Wiki10-31K datasets. Therefore, label text features may become a strong signal to characterize the relevance between the given document and a label. As shown in Table IV, compared with X-GRank-1 w/o LF, $P@1$ and $P@3$ of X-GRank-1 w/ LF increased by 0.35% and 1.11% on Eurlex-4K, which verified the positive effects of label features. In addition, when the label text feature is added to X-GRank-1 on Wiki10-31K, $P@1$ increased by 1.0%. Therefore, we added label text features as one of the inputs of X-GRank.

E. Performance on tail labels

We verified the performance of X-GRank on tail labels. The comparison results are shown in Table V. X-GRank almost completely outperformed AttentionXML on small-
scale datasets. In addition, the large-scale datasets, Wiki-500K and Amazon-670K, contains many tail labels. The average sample numbers per label of those datasets are only 16.86 and 3.99, and the tail labels account for 43.42% and 76.31% of all labels, respectively. Compared with AttentionXML, X-GRank has achieved better performance on all metrics except PSP\textsuperscript{0.5} on Wiki-500K. The experimental results verified the superiority of effectiveness of X-GRank on tail labels.

F. Evaluations of efficiency

We compare the efficiency of X-GRank with AttentionXML and LightXML of single model, marked with suffix of “-1”. The experiment is carried out under the same hardware with one Tesla V100 GPU. The input token lengths of AttentionXML, LightXML and X-GRank are set to 128. The results on the Wiki-500K and Amazon-670K datasets are shown in Table VI. Specifically, for the Amazon-670K dataset, the training time of X-GRank-1 is 8.44 ms per sample, which is lower than that of AttentionXML-1 and LightXML-1 with a remarkable reduction range of 34.06% and 40.14%. The reason why X-GRank-1 is faster than LightXML-1 is that we optimized the implementation of cluster retrieving of X-GRank-1, which reduces the data exchange between the CPU and GPU. The model size of X-GRank-1 is the smallest among the compared models, because the dimension of label embedding of X-GRank-1 is 128 while that of LightXML-1 is 400, which cuts down 182 million parameters. The experimental results verified the superiority of X-GRank in efficiency.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Metrics</th>
<th>AttentionXML-1</th>
<th>LightXML-1</th>
<th>X-GRank-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki-500K</td>
<td>Train Time (ms/sample)</td>
<td>7.5</td>
<td>6.9</td>
<td>8.71</td>
</tr>
<tr>
<td></td>
<td>Inference Time (ms/sample)</td>
<td>4.20</td>
<td>3.75</td>
<td>3.36</td>
</tr>
<tr>
<td></td>
<td>Model Size (GB)</td>
<td>3.11</td>
<td>1.47</td>
<td>0.714</td>
</tr>
<tr>
<td>Amazon-670K</td>
<td>Train Time (ms/sample)</td>
<td>12.8</td>
<td>14.1</td>
<td>8.44</td>
</tr>
<tr>
<td></td>
<td>Inference Time (ms/sample)</td>
<td>8.59</td>
<td>4.09</td>
<td>3.12</td>
</tr>
<tr>
<td></td>
<td>Model Size (GB)</td>
<td>5.52</td>
<td>1.53</td>
<td>0.797</td>
</tr>
</tbody>
</table>

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel GRank loss to solve the rank-missing and label imbalance problems in the existing work. We apply GRank loss to XMC problem and proposed a new model named X-GRank, which includes a label retrieving task and a label ranking task. The former is a cluster-granularity multi-label classification to generate the label candidate shortlist. The latter aims to rank the retrieved labels. Both tasks used the GRank loss to boost the results. X-GRank is trained in an end-to-end way. With extensive experiments, X-GRank shows high efficiency with the best precision on widely used large scale datasets, such as Wiki-500K and Amazon-670K, compared to the current state-of-the-art methods. In the future, we plan to apply the idea of group ranking to more scenarios, such as multi-label image classification, image and text matching, dense passage retrieval, which opens a new perspective for researchers in a wide range of applications.

REFERENCES

Interactive Visualization of Temporal Brain Connectivity Data based-on Frequent Feature Mining

Guangwei Zhang¹, Ming Jing¹, Yunjing Liu¹, Li Zhang¹,²,³*, Anming Dong¹,²,³, Jiguo Yu²,³

¹ School of Computer Science and Technology, Qilu University of Technology, Jinan, China
² Big Data Institute, Qilu University of Technology, Jinan, China
³ Shandong Fundamental Research Center for Computer Science, Jinan, China
* lizhang@qlu.edu.cn

Abstract—Medical data visualization is instrumental in assisting disease diagnosis and exploring brain function and structure. In this paper, we constructed a brain connectivity network using changes in BOLD signals at different time intervals and identified frequent characteristics to help doctors quickly pinpoint areas of interest. To study the changes in connectivity between brain regions, we visualize frequent sequences and compare them, highlighting important temporal features of patient brain areas. This makes the study and analysis of fMRI data more convenient and assists doctors in investigating abnormalities in the connections between brain functional areas.

Index Terms—Visual analysis, temporal graph comparison, brain connectivity data

I. INTRODUCTION

Functional magnetic resonance imaging (fMRI) is a brain imaging method commonly used to study cognition in the brain, which can explore the neural basis of cognition and emotion [1]. This approach produced a wealth of data that revealed patterns of hemodynamic activity throughout the brain. It plays an important role in describing functional connections in the brain under various clinical conditions.

Brain neurons are interconnected to form vast and complex networks, and fMRI contains a lot of data. Some researchers carry out feature extraction on the obtained data and apply the data to automatic diagnosis of brain diseases and other fields [2]. However, there are relatively few researches on the extraction of time series features, and no time attribute is added to the data for research. Graph models and mathematical tools are often used to describe and analyze network data. As the scale of network data increases, the visualization method changes from static graph to dynamic graph, and some visual metaphor methods also appear. Some researchers present link information by building abstract views, generating functional link matrices and anatomical views of the brain. And linking this information to anatomy helps neuroscientists locate spatial representations of brain regions [3]. But this analysis lacks comparisons of temporal features to help study changes in brain connectivity over time. We can see that there are few methods for feature extraction for a large number of data, and visual results are only represented by one view, which is inconvenient for data exploration and comparison.

In this work, we mined time sequence features from the constructed brain connection network, used the extracted time series data to generate dynamic graphs, and introduced a new visual analysis framework for comparison between dynamic graphs. The visualization results are placed in a Web-based GUI with an optimized nested comparison layout that allows you to compare timing data without affecting the data topology. For exploratory data analysis, we use linked views for visualization by interactively exploring the relationships between the different dimensions of the data set. Instead of relying on a priori model of complex relationships in high-dimensional data, researchers develop an understanding of the principles that describe the system.

In Section II, we review related research work. In Section III, we introduce application design ideas and datasets with temporal properties. In Section IV, we present the results of experiments on the data. In Section V, we conclude the paper and discuss future work.

II. RELATED WORK

The visualization of functional brain networks and feature extraction methods in brain networks are briefly described below.

In order to study the differences of brain networks in clinical studies, Gleicher et al. [4] proposed to overlay and display the adjacency matrix to facilitate visual comparison of data. However, due to the complexity of spatial structure changes and the extensibility of time dimension, a visual coding may not be suitable for all visualization tasks. The above methods are combined with the adjacency matrix visualization method to study the brain connection data. The multi-view hybrid visualization method can balance the advantages and disadvantages of a single visualization technology. "In Situ” supports a variety of visualization techniques such as vertex link graphs, adjacency matrices, and timelines to maximize insight and avoid misunderstandings [5]. The connection view

DOI reference number: 10.18293/SEKE2023-167
is adopted in the research [6], which can interactively explore
the relationship between different dimensions of the data set.
This method allows researchers to explore and analyze the
data. Therefore, we use the adjacency matrix overlapping to
visualize the time series. Multiple views were used to explore
and analyze changes in brain functional connectivity.

While advances have been made in disease prediction and
other applications due to the development of AI technology,
how to detect and validate specific brain connectivity features
remains challenging. Yang et al. [7] proposed a comprehensive
analysis method to compare blocky brain networks. Different
from the previous methods, the researchers combined the two
visualization methods of node-connection graph and adjacency
matrix, used the clustering algorithm to divide the brain
regions, and improved NodeTrix [8] to design and display
the block-level brain network connection mode. On the basis
of the above research, feature extraction is performed on the
constructed brain network, and frequently changing regions
are obtained to help amateurs quickly locate the brain regions
of interest.

III. MATERIALS AND METHODS

In this study, we designed a Web-based interactive visual
interface (Fig. 2). The method is expected to help researchers
analyze brain connectivity data, allowing them to compare
differences between samples from multiple dimensions.

A. Dataset

In this study, time series data of brain functional connectiv-
ity were obtained by processing ADHD-200 data set. The two
groups of data were respectively from children with normal
brain function development (TDC for short) and children with
ADHD. Subjects were all in a resting state when fMRI data
was collected. The processed time series correlated with brain
regions. Each set of data, combined with the AAL template,
was divided into 116 anatomical regions containing 172 time
series.

B. Feature Extraction

The connectivity features between brain regions are very
high dimensional, and it is expensive to verify many comput-
utional models proposed in complex brain network studies.
Therefore, we use machine learning technology to extract
features from brain connectivity data. In the frequent sequence
screening process, the same step size is set to divide the data
into a certain number of time segments, the BLOD signal
changes of nodes in adjacent time segments are calculated,
and then the nodes that have asynchronous changes of blood
oxygen concentration in the same time period and meet the
linear correlation are found. Arrange the matrix in order
of time segments to form a network of brain connections.
Next, we look for nodes that occur more frequently in the
brain connectivity network, and extract the previous node
and the last node of the node to form frequent sequences.
As to whether the extracted features are discriminant or not,
we classified the extracted features, and the results showed
that there were significant differences in feature distribution
between normal people and patients.

In order to verify whether the extracted features are dis-
riminant, SVM was used to classify the extracted features.
The results showed that there were significant differences in
feature distribution between normal subjects and patients.

C. Visual Design

1) Colorwall: We do a motion picture comparison in the
Colorwall area. In this view we filter the data based on
embedded visualization techniques and also add a design to
assist comparison. For data visualization, choose a method
where two squares overlap. By choosing the same hue on
both squares, the user can easily compare the mapping values
between the squares by observing only the difference in
brightness between the inner and outer areas of the squares.
Because of the time series data, we use the arc of the curve
to represent the direction of time(Fig. 1).

As dynamic networks become more complex, we need to
design the layout. Grouping nodes visually simplifies the page
and facilitates the analysis of the data. In order to ensure that
the visual dynamic graph can avoid visual confusion, we group
the time series data. Nodes in a group are deployed in Force
mode to prevent node overlap. Layout allows users to intervene
and adjust, further preventing overlapping of data visualization
results.

2) Anatomy: In this panel, we generate anatomical images
of the brain, and we can study and analyze the connectivity
between brain regions in three dimensions. We processed the
time sequence data to generate a functional connection matrix,
and drew a 3d brain corresponding to the aal template. A dot
is drawn to mark the corresponding location of each functional
area, and the lines between functional areas indicate the degree
of correlation between brain functional areas. Part of the region
connection is based on the clustering results in Colorwall
region.
Attention-deficit hyperactivity disorder (ADHD) is one of the most common developmental disorders in children, and children often have symptoms such as impulsiveness, inattention and hyperactivity [9], [10]. In recent years, cognitive functional magnetic resonance imaging (fMRI) and resting-state fMRI have been widely used to examine abnormal brain function. The detection of abnormal conditions in the resting state can be applied to mental and developmental disorders such as ADHD, Alzheimer’s disease and depression.

In this study, we were able to select the regions of the brain we were interested in and get the visualizations. Among them, the related regions of the cerebellum are one of the most different regions in our feature mining results. The visualized results were obtained by selecting the functional brain regions mentioned above, as is shown in Fig[3] Number 110 corresponds to the Vermis_3 area in the AAL template. It can be seen from the colorwall region in the figure that there is connectivity between this region and the other four regions, and there is a large connectivity difference between patients and normal people. Taking the time sequence data between Vermis_1_2 (No. 109) and vermis_3 (No. 110) as an example, the time direction is from vermis_3 to Vermis_1_2. In the process of time change, ADHD patients were connected at the first four time points, while TDC were more connected at the later time points. In the Plot area, by comparing the trend curve and observing the difference curve, we can find that there are also large differences in values between samples. We can intuitively see the difference in BOLD signal between patients and normal people(Fig[4]). This difference can also be identified simultaneously in the anatomical view, which shows that functional connectivity in the brains of ADHD patients is generally lower than that of TDC. As described in reference [11], cerebellar functional connectivity disorders may be one of the pathological and physiological causes of ADHD cognitive impairment. At the same time, we can find that most of the connected regions in this group are located in the cerebellum. Through the comparison of multiple samples, it is found that the cerebellar regions in the division of brain network are often in the same cluster. Moreover, there were significant connectivity differences between ADHD patients and TDC in the cerebellar region.

### IV. RESULTS

### V. CONCLUSION AND FUTURE WORK

In this study, we mainly designed an application for visual comparison of fMRI data. By processing the time series data generated by fMRI for visualization, we can study the changes of dynamic graphic data and highlight the difference between two sets of different data. We use AI technology to classify brain diseases, mine frequent features, highlight important time series features in patients’ brains, and assist doctors in diagnosing diseases. Our work then suffers from some deficiencies in the handling of complex networks, manifested in the removal of excessive features to simplify the content, and the possibility of obscuring highlighted points by connecting lines in anatomical views. In the future, optimized algorithms will be used to reasonably filter temporal information and better visualize the connectivity of anatomical views.

### VI. ACKNOWLEDGMENT

This work was supported by the “Colleges and Universities 20 Terms” Foundation of Jinan City, China (202228093), the Piloting Fundamental Research Program of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2022XDD001, the National Natural Science Foundation of China (61902202), and International Cooperation Foundation of Qilu University of Technology (45040118).

### REFERENCES

Fig. 2: Interactive interface overview.

(a) Colorwall
(b) ADHD
(c) TDC
(d) Plot

Fig. 3: (a) Comparison of the dynamic graphs of ADHD patients and TDC. (b) and (c) are 3D brain maps of ADHD patients and TDC, respectively. (d) Plot.
A Dynamic Drilling Sampling Method and Evaluation Model for Large-Scale Streaming Data

Peng Zhang, Zhaohui Zhang*, Chaochao Hu, Pengwei Wang
School of Computer Science and Technology
Donghua University
Shanghai 201600, China
zhzhang@duhu.edu.cn

Abstract—The sampling method for real-time and high-speed changing streaming data is prone to lose the value and information of a large amount of discrete data, and it is not easy to make an efficient and accurate streaming data valuation. The SDSLA (Streaming Data Drilling Sampling Method Under Limited Access) sampling method based on mineral drilling exploration can streamline data valuation containing many discrete data in real-time, but when the range of discrete data in streaming data is irregular, it has low sampling accuracy for discrete data. Based on the SDSLA algorithm, we propose a dynamic drilling sampling method SDDS (Streaming Data Dynamic Drilling Sampling). This method takes well as the analysis unit dynamically changes the size and position of the well, and accurately predicts the position and range of discrete data. A new model SDVEM (Streaming Data Value Evaluation Model), is further proposed for data valuation, which evaluates the sample set from discrete, centralized, and overall dimensions. Experiments show that the method proposed in the paper uses neural network training and testing with a small sampling rate to obtain accuracy, recall, and F1 scores above 90%, which is higher than that of the SDSLA algorithm. In summary, the SDDS sampling method is beneficial to the training neural network models and evaluating the value characteristics of streaming data, which has essential research significance in big data valuation.

Keywords—streaming data; dynamic drilling; data valuation; neural network

I. INTRODUCTION

In the era of big data, the data valuation is one of the most essential requirements of big data. C.R. Yin et al. [1] proposed data value can be considered a new form of value, and how to evaluate its value has become a new problem. For big data valuation, A.F. Haryadi et al. [2] proposed more than half of financial services organizations report that big data is not delivering the expected value. Big data usually exists in the form of streaming data, and it has the characteristics of high-speed and real-time, which brings severe challenges to sampling high-value data. Therefore, methods for sampling and streaming data valuation are needed.

Sampling is a required data analysis method in big data, and it plays an irreplaceable role in big data valuation. Currently, sampling methods for streaming data are mainly divided into two categories. The first category is unbiased sampling methods: stratified sampling [3], random sampling [4], reservoir sampling [5]. Unbiased sampling is random, and the streaming data obtained by sampling will lose some key information. The second category is biased sampling methods: probability density sampling[6]. Biased sampling can preserve many discrete data in streaming data but amplifies the impact of discrete data in the sample set.

To sum up, there are still some difficulties: 1. How to accurately predict the position and range of discrete data in streaming data? 2. Too much discrete data in the sample is not easy to use to evaluate the overall characteristics of streaming data. Given the above problems, the contributions of the paper are summarized as follows:

- We propose a dynamic sampling method SDDS, which takes well as the analysis unit dynamically changes the size and position of the well, and accurately predicts the position and range of discrete data.
- We propose a new streaming data valuation model SDVEM which evaluates the sample set from discrete, centralized, and overall dimensions.

Section II introduces related work. Section III depicts the SDDS method. Section IV depicts the streaming data valuation model. Section V describes the dataset and experimental analysis. Section VI summarizes the research results of the paper and expectations.

II. RELATED WORK

At this stage, some sampling methods can collect discrete data so that the sample set can be used as the training set of the neural network. T. Li et al. [7] proposed a new data transformation method, the KSB algorithm, that improved machine learning models’ performance.

Assessing the value of data, P. J [8] pointed out how we can objectively, systematically, and quantitatively assess the value of data. F.J. Xu et al. [9] proposed a streaming data drilling sampling method (SDLSS) and an overall feature evaluation model of streaming data sets. The main idea of the SDLSS method is to use the skewness coefficient to locate the following well-drilling position, to perform in-well sampling and inter-well sampling respectively. The disadvantage is that when the range of discrete data in streaming data is irregular, the sampling accuracy of discrete data is low. Based on the SDLSS method, we propose a complete SDDS method which dynamically changes the size of the wells for predicting the range of discrete data and changes the size of the well-interval...
adaptively. Experiments show that the sampling accuracy of the SDDS method in discrete data is higher than that of SDLSA, the effect of evaluating the SDDS sample set from three dimensions of discrete, centralized, and overall is better than that of SDLSA, and the training effect of the sample set of SDDS for neural network models is better.

### III. Dynamic Drilling Sampling Method

Definition 1 (Streaming Data): The streaming data $S$ is represented as:

$$S = \{(id_i, time_i, value_i)|1 \leq i \leq N \text{ and } i \in N^+\} \quad (1)$$

Where $id_i$ is the order of the $i^{th}$ data, $time_i$ is the arrival time of the $i^{th}$ data, $value_i$ is the size of the $i^{th}$ data, $N$ is the size of streaming data. An example of $S$ is shown in Figure 1.

Definition 2 (Well): We use well as analysis units. The $i^{th}$ well $W_i$ is expressed as:

$$W_i = \{(id_j, time_j, value_j)|1 \leq j \leq WS_i \text{ and } j \in N^+\} \quad (2)$$

Where $WS_i$ is the size of the $i^{th}$ well.

Definition 3 (Well-Interval): The $i^{th}$ well interval $WI_i$ is expressed as:

$$WI_i = \{(id_j, time_j, value_j)|id_{wi,max} + 1 \leq id_j \leq id_{wi+1,min} - 1\} \quad (3)$$

Where $id_{wi,max}$ is the largest id in the $i^{th}$ well, $id_{wi+1,min}$ is the smallest id in the $i + 1^{th}$ well.

![Figure 1. Streaming data schematic diagram](image)

We propose the SDDS sampling method to obtain valuable discrete data in the streaming data. Firstly, setting the initial well and initial well-interval, k-means clustering [10] in the well. Secondly, calculating each class's sampling rate and well-interval by deviation coefficient. Thirdly, using intra-class unbiased sampling and inter-class biased sampling in the well, sampling equidistant in the well-interval. Finally, predicting the size of the following well through the correlation coefficient and the coefficient of variation. The specific sampling method is shown in Figure 2.

![Figure 2. The large-scale streaming data sampling](image)

A. Dynamic Adjust Sampling Rate And Well-Position

To accurately predict the position of discrete data in the streaming data. Firstly, we use the k-means clustering to divide well-data into three categories, then carry out intra-class unbiased and inter-class biased sampling, increasing the sampling rate of the minority class. Assuming that the current is the $i^{th}$ well, the initial sampling rate is $p_{init}$, and the adjustment formula of the sampling rate is as follows:

$$p = \left\{ \begin{array}{ll} 2 \times p_{init} \times |SK_i|, & SK_i \in (-0.5,0.5) \text{ or } (0.5,1) \\ 2 \times p_{init} \times |SK_i|, & SK_i \in (-\infty,-1) \text{ or } (1,\infty) \end{array} \right. (4)$$

If $SK_i \in [-0.5,0.5]$, the sampling rate for all classes $p = p_{init}$; if $SK_i \in (-\infty,-1) \text{ or } (1,\infty)$, increase the sampling rate of the two classes with fewer numbers $p = 2 \times p_{init} \times |SK_i|$; if $SK_i \in (-1,0.5) \text{ or } (0.5,1)$, increase the sampling rate for the least number of classes $p = 2 \times p_{init} \times |SK_i|$.

Secondly, we use the deviation coefficient [11] to adjust the well-interval size dynamically, and the size of $i^{th}$ well-interval formula $WIS_i$ is as follows:

$$WIS_i = \left\{ \begin{array}{ll} WIS_{init}, & SK_i \in (-\infty,-1) \text{ or } (1,\infty) \text{ or } (-1,0.5) \text{ or } (0.5,1) \end{array} \right. \quad (5)$$

Where $WIS_i$ stands for the $i^{th}$ well-interval size, $SK_i$ stands for the deviation coefficient of the $i^{th}$ well.

Specifically divided into two situations, if $SK_i \in [-0.5,0.5]$, increase well-interval $WI_i = 2 \times WIS_{init}$; if $SK_i \in (-\infty,-1) \text{ or } (1,\infty) \text{ or } (-1,0.5) \text{ or } (0.5,1)$, reduce well-interval $WI_i = [WIS_{init}]/|SK_i|$.  

B. Dynamic Adjust The Well Size

To accurately predict the range of discrete data in the streaming data. Firstly, we define the peaks and troughs of streaming data through three features: 1. The slope changes very large; 2. Periodic changes; 3. The degree of dispersion is low. As shown in Figure 3(a), the peaks are divided into SP(Shock-Peak), OP(Oscillation-Peak), and BP(Buffer-Peak). As shown in Figure 3(b), the troughs are divided into ST(Shock-Trough), OT(Oscillation-Trough), and BT(Buffer-Trough). Where the SP and ST have a huge slope and the degree of dispersion will be huge; the OP and OT have periodic changes and the degree of dispersion will be relatively large; BP and BT have the lowest degree of dispersion of the well. When two different wells contain the same kind of peak or trough, the two wells have a certain self-similarity, and the degree of dispersion of the two wells will be very close.

![Figure 3. Streaming data peak and trough classification diagram](image)
Secondly, we propose an AWS algorithm (Adaptive well sizing) to predict the range of discrete data in streaming data accurately, combining Pearson correlation coefficient [12] and variation coefficient [13]. The AWS records the representative wells in the well-set, uses the sliding window to accept the newly arrived data, traverses the well-set, sets the sliding window size to the size of the different wells in the well-set, and sets the size of the next well to the one with the highest correlation coefficient with the sliding window in the well-set. The specific algorithm is as follows:

Algorithm AWS: Adaptive Well Size

Input: \(W_{init}\) - Init Well; \(\delta\) - Threshold Value of Correlation Coefficient; \(WC\) - Well Collection; \(SW\) - Sliding Window; \(PCC\) - Pearson Coefficient Collection.

Output: \(WS\)

1. \(WC.add(W_{init})\)
2. for data in \(WC\):
3. \(SW.clear()\)
4. \(SW.size = data.size\)
5. \(SW.add(SW.size)\)
6. \(PC = Pearson\ correlation\ coefficient\ of\ data\ and\ SW\)
7. \(PCC.add(PC)\)
8. \(PC\_MAX = max(PCC)\)
9. if \(PC\_MAX \geq \delta\):
10. \(index = PCC.index(PC\_MAX)\)
11. \(WS = WC[index].length\)
12. \(WC.add(Latest\ well-data)\)
13. else:
14. \(SW.clear()\)
15. \(SW.size = data.size\)
16. \(SW.add(SW.size)\)
17. \(SW\_COV = Variation\ coefficient\ of\ SW\)
18. \(AWC = Covariance\ of\ all\ wells\)
19. if \(SW\_COV \geq 75\% AWC\):
20. \(WS = 2 \times W_{init}.size\)
21. elif \(SW\_COV \geq 50\% AWC\):
22. \(WS = 1.5 \times W_{init}.size\)
23. elif \(SW\_COV \geq 25\% AWC\):
24. \(WS = W_{init}.size\)
25. else:
26. \(WS = W_{init}.size / 2\)
27. \(WC.add(Latest\ well-data)\)
28. return \(WS\)

C. Dynamic Sampling Algorithm

We propose a SDDS algorithm to accurately predict the position and range of discrete data. Firstly, setting the initial well and initial well-interval, k-means clustering in the well. Secondly, calculating each class's sampling rate and well-interval by deviation coefficient. Thirdly, using intra-class unbiased sampling and inter-class biased sampling in the well, sampling equidistant in the well-interval. Finally, predicting the size of the following well through the AWS algorithm.

Algorithm SDDS: Streaming Data Dynamic Sampling

Input: \(WS_{init}\) - Init Well-Size; \(S\) - Streaming Data; \(WS_{init}\) - Init Well Interval; \(p_{init}\) - Init Sampling Rate

Output: \(SS\) - Sample Set

1. Set \(W_{init}.length = WS_{init}\)
2. Set \(WI_{init}.length = WS_{init}\)
3. K-means clustering for \(W_{init}\), get three classes of data
4. Calculate the \(p\) for each class in \(W_{init}\), according to formula (4)
5. Sample by steps 13–23 of this algorithm and add to \(SS\)
6. Equidistant sampling of well interval data
7. while \(S\) is generating:
8. Get the size of the next well by AWS algorithm: \(WS_{next}\)
9. Set \(WI_{next}.length = WS_{next}\)
10. K-means clustering for \(W_{next}\), get three classes of data
11. Calculate the \(p\) of three classes in \(W_{next}\) by formula (4)
12. if \(\delta\) is 
13. Sampling rate for all classes is \(p_{init}\)
14. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
15. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
16. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
17. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
18. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
19. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
20. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
21. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
22. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
23. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
24. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
25. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
26. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
27. \(SS.add(Data\ obtained\ from\ W_{next}\ by\ reservoir\ sampling)\)
28. return \(SS\)

IV. SDVEM Evaluation Model

We propose SDVEM evaluation model to streaming data valuation. The specific model is shown in Figure 4.

Figure 4. SDVEM Evaluation Model

A. Discrete Dimension Evaluation Sample Set

The discrete data of the raw streaming dataset \(DDRD\) and the discrete data of the sample set \(DDSS\) are as follows:

\[
DDRD = \{RD_i\} \quad \text{and} \quad \text{where} \quad RD_i \geq RD \times \delta_{upper} \lor RD_i \leq RD \times \delta_{down}\}
\]

\[
DDSS = \{SS_i\} \quad \text{and} \quad \text{where} \quad SS_i \geq SS \times \delta_{upper} \lor SS_i \leq SS \times \delta_{down}\}
\]

Where \(RD_i\) is the \(i^{th}\) data of raw streaming data, \(\delta_{upper}\) and \(\delta_{down}\) are the threshold to decide whether it is discrete data, \(\overline{RD}\) is the mean of raw streaming data, \(\overline{SS}\) is the \(i^{th}\) of the sample set, and \(\overline{SS}\) is the mean of the sample set.

Definition 4 DMA (Discrete Mean Accuracy): refers to the accuracy rate of estimating the mean value of the \(DDRD\) attribute value with the mean value of the \(DDSS\) attribute value. The formula of DMA is as follows:
\[ DMA = 1 - \left( \frac{\text{DDR} - \text{DDSS}}{\text{DDR}} \right) \times 100\% \]  

Definition 5 ADCV (Accuracy of Discrete Coefficient of Variation): refers to the accuracy rate of estimating the coefficient of variation of the DDR attribute value by using the coefficient of variation of the DDSS attribute value. The formula of ADCV is as follows:

\[ ADCV = 1 - \left( \frac{\text{CV}_{\text{DDR}} - \text{CV}_{\text{DDSS}}}{\text{CV}_{\text{DDR}}} \right) \times 100\% \]  

Definition 6 DSA (Discrete Sampling Accuracy): refers to the ratio of the intersection number of the DDSS attribute value and the DDRD attribute value to the DDSS length. The formula of DSA is as follows:

\[ DSA = \frac{\text{len}(\text{DDSS} \cap \text{DDR})}{\text{len}(\text{DDSS})} \times 100\% \]  

B. Centralized Dimension Evaluation Sample Set

The centralized data of the raw streaming dataset CDRD and the centralized data of sample set CDSS are as follows:

\[ \text{CDRD} = \{ \text{RD} | \text{RD} \times \delta_{\text{down}} \leq \text{RD} \leq \text{RD} \times \delta_{\text{upper}} \} \]
\[ \text{CDSS} = \{ \text{SS} | \text{SS} \times \delta_{\text{down}} \leq \text{SS} \leq \text{SS} \times \delta_{\text{upper}} \} \]

Definition 7 CMA (Centralized Mean Accuracy): refers to the accuracy of estimating the mean value of the CDRD with the mean value of the CDSS. The calculation formula of CMA is as follows:

\[ CMA = 1 - \left( \frac{\text{CDRD} - \text{CDSS}}{\text{CDRD}} \right) \times 100\% \]

Definition 8 ACCV (Accuracy of the Centralized Coefficient of Variation): refers to the accuracy of estimating the coefficient of variation of the CDRD with the coefficient of variation of the CDSS. The calculation formula of ACCV is as follows:

\[ ACCV = 1 - \left( \frac{\text{CV}_{\text{CDRD}} - \text{CV}_{\text{CDSS}}}{\text{CV}_{\text{CDRD}}} \right) \times 100\% \]

C. Overall Dimension Evaluation Sample Set

Definition 9 OMA (Overall Mean Accuracy): refers to the accuracy of estimating the original streaming data mean by using the sample set mean. The calculation formula of OMA is as follows:

\[ OMA = 1 - \left( \frac{\text{RD} - \overline{\text{SS}}}{\text{RD}} \right) \times 100\% \]  

Where RD is the mean of the raw streaming data, SS is the mean of the sample set.

Definition 10 AOCV (Accuracy of the Overall Coefficient of Variation): refers to the accuracy of estimating the coefficient of variation of raw streaming data with the coefficient of variation of sample set. The calculation formula of AOCV is as follows:

\[ AOCV = 1 - \left( \frac{\text{CV}_{\text{RD}} - \text{CV}_{\text{SS}}}{\text{CV}_{\text{DDR}}} \right) \times 100\% \]

V. EXPERIMENTS AND ANALYSES

In this section, we introduced the dataset, analyzed the impact of the SDVEM model on different parameters, and compared the SDDS algorithm with the SDSLA algorithm.

### A. Experimental Dataset

<table>
<thead>
<tr>
<th>DataSet</th>
<th>Data Volume</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSI</td>
<td>87645</td>
<td>0</td>
</tr>
<tr>
<td>NEWS</td>
<td>276336</td>
<td>0</td>
</tr>
<tr>
<td>HSII</td>
<td>55442</td>
<td>3</td>
</tr>
<tr>
<td>HSII2</td>
<td>55736</td>
<td>3</td>
</tr>
</tbody>
</table>

As shown in TABLE I, we selected four datasets HSI, NEWS, HSII and HSII2 to verify the effectiveness of the SDDS algorithm, among them HSI and NEWS are real datasets, HSII and HSII2 are synthetic datasets. In addition, we according to the degree of dispersion of experimental dataset, \( \delta_{\text{up}} \) is set to 1.5, and \( \delta_{\text{down}} \) is set to 0.5.

### B. Influence Analysis of Different Parameters

To prove that the AWS algorithm can predict the position and range of discrete data, the sample set obtained by the SDDS algorithm can preserve the discrete data and reflect the centralized and overall characteristics of the original streaming data. We use the SDVEM model to perform experimental evaluations on HSI and NEWS datasets under different parameters, a detailed analysis of parameter \( \delta \) in the AWS algorithm and parameters \( W_{\text{SI}} \), \( W_{\text{SI}} \) and \( p_{\text{SI}} \) in the SDDS algorithm. The larger the parameter \( \delta \), the higher the similarity standard between wells. Parameters \( W_{\text{SI}} \) and \( W_{\text{SI}} \) should be set based on the distribution characteristics of discrete data in the streaming data. The larger the parameter \( p_{\text{SI}} \), the smaller the proportion of discrete data in the sample set.

Firstly, \( W_{\text{SI}} \), \( W_{\text{SI}} \), and \( p_{\text{SI}} \) are set as 20, 20, and 0.1, respectively, the parameter \( \delta \) in the AWS algorithm was adjusted differently. The experimental results are shown in TABLE II.

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA</td>
<td>94</td>
<td>93</td>
<td>90</td>
<td>91</td>
<td>87</td>
</tr>
<tr>
<td>ADCV</td>
<td>92</td>
<td>94</td>
<td>96</td>
<td>94</td>
<td>95</td>
</tr>
<tr>
<td>DSA</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>CMA</td>
<td>94</td>
<td>94</td>
<td>92</td>
<td>92</td>
<td>91</td>
</tr>
<tr>
<td>ACCV</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OMA</td>
<td>79</td>
<td>69</td>
<td>71</td>
<td>71</td>
<td>72</td>
</tr>
<tr>
<td>AOCV</td>
<td>90</td>
<td>89</td>
<td>87</td>
<td>89</td>
<td>88</td>
</tr>
</tbody>
</table>

Secondly, \( \delta \), \( W_{\text{SI}} \), and \( p_{\text{SI}} \) are set as 0.2, 20, and 20, respectively, different adjustments are made to the parameter \( W_{\text{SI}} \) in the SDDS algorithm, and the experimental results are presented in TABLE III.
The characteristics of the SDDS algorithm can reflect the almost all above 85%. It proves that the sample set obtained by the almost all above 90%, and the evaluation accuracy of AOCV in the discrete and centralized dimension is almost all above 96%. The experimental results are shown in TABLE IV and TABLE V.

Thirdly, $\delta$, $W_{\text{init}}$, and $p_{\text{init}}$ are set as 0.2, 10, and 0.1, respectively, different adjustments are made to the parameter $W_{\text{init}}$ in the SDDS algorithm, and the experimental results are presented in TABLE IV.

Fourthly, $\delta = 0.2$, $W_{\text{init}} = 20$, and $W_{\text{init}} = 20$ are set as 0.2, 20, and 20, respectively, and different adjustments are made to the parameter $p_{\text{init}}$ in the SDDS algorithm, and the experimental results are exhibited in TABLE V.

It can be seen from TABLE I, TABLE II, TABLE III, TABLE IV and TABLE V that the evaluation accuracy of the SDDS algorithm in the discrete and centralized dimension is almost all above 90%, and the evaluation accuracy of AOCV is almost above 85%. It proves that the sample set obtained by the SDDS algorithm can reflect the discrete, centralized and overall characteristics of the raw streaming data.

C. Comparison of Experimental Results of Streaming Data Value Evaluation Model

To prove that the SDDS algorithm can evaluate the value of streaming data from three dimensions: discrete, centralized, and overall. We use real data sets HSI and NEWS to conduct comparative experiments on SDDS and SDSSLA sampling. The experimental results are shown in Figure 5 and Figure 6.
It can be seen from Figure 5 and Figure 6 that the evaluation accuracy of the sample set obtained by the SDDS algorithm is very high in the discrete, centralized, and overall dimensions, and the accuracy of the five evaluation indicators ADCC, DSA, CMA, ACCV, and AOCV almost both are above 90%, the evaluation accuracy of DMA is almost above 85%, and the JSD is also very low, indicating that the probability distribution of the sample set and the original streaming dataset is very close. Compared with the SDL NSA algorithm, the evaluation accuracy of the SDDS algorithm is almost higher in the three dimensions discrete, centralized, and overall.

D. Comparison of Neural Network Training Effect

To prove that the sample set obtained by the SDDS algorithm is beneficial to reduce the amount of data required for neural network model training. We use HSI1 and HSI2 to divide the dataset into 80% training set and 20% test set, then samples the training set, and conducts comparative experiments on the datasets before and after sampling.

Based on the above experiments, we select parameters with the best evaluation result, setting δ, Wsinit, Wsinit as 0.2, 20, and 20, respectively, different adjustments were made to the parameter pinit in the SDDS algorithm, the experimental results are shown in Figure 7.

![Figure 7. Training F1, Recall and Accuracy values of raw streaming data, the SDSLA and SDDS sample sets on the HSI1 and HSI2.](image)

It can be seen from Figure 7 that the F1 value and recall rate of the neural network model trained with the sample set obtained by the SDDS algorithm are almost higher than 90%, and the accuracy is almost higher than 95%, which is almost the same as using the original streaming dataset training. Moreover, the sample set obtained by the SDDS algorithm is better than the SDL NSA algorithm in the effect of training the neural network model.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose the SDDS algorithm to predict the position and range of discrete data and obtain a sample set containing more discrete data. It can describe the original stream data set's value characteristics and be well used in training neural network models. The streaming data value evaluation model SDVEM fully and detailedly analyzes the sample set from three dimensions: discrete, centralized, and holistic, which has essential research significance for the value evaluation of streaming data in the field of big data. In future work, we will further increase the dimension of value evaluation of streaming data and more comprehensively evaluate the value of streaming data.

ACKNOWLEDGMENT

This work was supported in part by the Shanghai Science and Technology Innovation Action Plan Project (22511100700).

REFERENCES


ASTD: Automatic Seasonal-Trend Decomposition for Time Series

Bowen Chen\textsuperscript{1,2}, Hancheng Lu\textsuperscript{2}, Yuang Chen\textsuperscript{2}, Haoyue Yuan\textsuperscript{2}, and Minghui Wang\textsuperscript{3}
\textsuperscript{1} Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei, China
\textsuperscript{2} University of Science and Technology of China, Hefei, China
\textsuperscript{3} AI Institute of H3C Technologies Co., Ltd.,
{chenbowen, yangchen21, yhyue}@mail.ustc.edu.cn, hclu@ustc.edu.cn, mhwang@h3c.com

Abstract—The rapid and accurate decomposition of multi-period time series is crucial for reliable forecasting, anomaly detection, and classification. However, the traditional approach of first detecting the periodicity and then selecting from a range of decomposition algorithms based on the periodicity results in inefficiencies and complexity. To address this challenge, we propose the automatic seasonal-trend decomposition (ASTD), a unified method for automatic time series decomposition. With the ASTD, users no longer need to worry about whether their time series is multi-period, single-period, or aperiodic. They simply provide the time series, and the ASTD automatically returns the final decomposition results. Careful consideration of runtime cost and accuracy requirements has been taken in the design of the ASTD, which has an overall time complexity of $O(N\log N)$. Extensive experimental results show that the proposed ASTD outperforms other state-of-the-art decomposition algorithms in terms of minimum mean square error (MSE) and mean absolute error (MAE). Notably, when applied to the Taylor dataset, the ASTD is approximately 3 times faster than other baseline decomposition algorithms.

Index Terms—time series, periodic detection, seasonal-trend decomposition

I. INTRODUCTION

The digital age has led to an increase in recorded time series, such as Central Processing Unit (CPU) usage and request numbers in Operation and Maintenance, spending and purchasing indices in economics, and daily electricity consumption. Many time series have periodic behavior, such as human heartbeat and rush-hour traffic, as well as tides related to lunar and solar cycles. Accurate time series decomposition is essential for tasks like anomaly detection [1]–[4], forecasting [5]–[8], and classification [9]–[12], and can improve prediction accuracy by extracting seasonal and trend components. Many decomposition algorithms have been proposed for time series, including single-period and multi-period algorithms. Single-period algorithms include STL [13] and RobustSTL [14], while multi-period algorithms include MSTL [15], Fast RobustSTL [16], STR [17], and TBATS [18]. Non-periodic time series can have their trend extracted using filters like HP [19], Supermoother [20], and l1 [21] trend filter. If unsure of the type of time series, beginners can use periodic detection tools to identify the periodicity and choose the corresponding algorithm.

There are two types of periodic detection tools: single period and multiple periods. Nevertheless, existing algorithms can be affected by outliers, noise, and spectral leakage, which can result in inaccurate periodicity detection and decomposition. Auto-period [22] may select incorrect periods due to spectral leakage and limited resolution. CFD-Autoperiod [23] addresses this issue with clustering, filtering, and detrending, but noise can still cause the centroid to be offset from the actual period. Robust-period [24] uses Huber-periodogram and Huber autocorrelation function (Huber-ACF) to mitigate outlier effects, but it requires multiple iterations, making it time-consuming.

Seasonal-trend decomposition methods have been proposed to analyze periodic information, but STL is only suitable for single-period time series. MSTL was introduced in 2021 to handle multi-period time series, but it may include data from other seasons. Hyndman proposed STR and TBATS for robustness to outliers in multi-period time series, but they have high computational complexity. Robust-STL was introduced to handle outliers, noise, and abrupt trend changes, but it is not suitable for multi-period time series. To address this limitation, Fast Robust-STL was introduced, which extends Robust-STL and reduces computational complexity, but still requires multiple iterations. Existing methods for time series only focus on either periodic detection or decomposition, making it inconvenient and inefficient for users.

To address these issues, we propose a unified method called ASTD for automatic time series decomposition, which detects and decomposes time series regardless of their periodic characteristics. ASTD uses a pruning method and partition idea to detect the period robustly and accurately, while a comprehensive score is calculated to mitigate spectral leakage effects. For decomposition, ASTD uses the input data spectrum to extract trends and seasons and the spectral residual (SR) [25] to obtain complete and accurate residuals. ASTD greatly expands its application in long time series and speeds up decomposition.

Our main contributions are listed as follows:

- We propose ASTD, a unified method for the automatic decomposition of periodic time series. With ASTD, users are no longer required manually detect the periodicity of time series and then select different decomposition algorithms according to the detection results.
- In ASTD, we propose a low-complexity and high-accuracy algorithm for detecting periodicity in multi-
period time series, which effectively mitigates the effects of noise and limited resolution of DFT.

- Additionally, we also propose a novel method for fast and accurate decomposition of multi-period time series, which can accurately quantify each component and extract them in a single calculation, without the need for multiple iterations.
- Finally, our proposed method is validated through experiments on real datasets, which demonstrate its generality and effectiveness.

The remainder of this article is structured as follows: Section II presents the framework of ASTD, Section III reports the experimental results and analysis, and Section IV concludes the paper.

II. FRAMEWORK OF ASTD

A. Framework Overview

ASTD has two main parts: periodic detection and decomposition, as shown in Figure 1. It aims to automatically decompose periodic time series accurately and efficiently. To achieve this, ASTD prunes invalid periods, uses blocking to reduce time complexity, and applies a seasonal trend decomposition algorithm with only one iteration. The periodic time series is defined as:

\[ X_t = T_t + \sum S_{i,t} + R_t, \quad t = 1, 2, \ldots, N \]  

(1)

where \( X_t \) is the observation at time \( t \), \( T_t \) is the trend, \( S_i \) is the sum of seasonal components with periods, and \( R_t \) is the residual signal.

B. Precise Period Detection

Real-world time series have diverse trends that can impact periodic detection accuracy. LOWESS (Locally Weighted Scatterplot Smoothing) is used to estimate trends, particularly for complex trends with no theoretical models.

Once the trend is estimated, the Periodogram and ACF are computed for the detrended time series. This involves working with a detrended sequence \( x' \).

\[ A(f) = \text{Amplitude}(\mathcal{F}(x')) \]  

(2a)

\[ P(f) = \frac{A^2(f)}{n} \]  

(2b)

\[ ACF(p) = \mathcal{F}^{-1}(P) \]  

(2c)

The Fourier Transform and Inverse Fourier Transform are denoted as \( \mathcal{F} \) and \( \mathcal{F}^{-1} \), respectively. The amplitude spectrum is denoted as \( A(f) \), the Periodogram as \( P(f) \), and the auto-correlation coefficient as \( ACF(p) \).

As a result of spectrum leakage, a periodogram may produce multiple potential periods for a signal. Relying solely on the power of the periodogram to determine the candidate period is not reliable. To achieve greater accuracy, the ACF can be utilized in conjunction with the periodogram to provide a more refined estimate of the candidate period. There are two steps involved in Period Detection, which we will explain in detail in the following sections:

- **Step 1: Screening.** To identify potential periods, we first set a threshold power equal to \( \lambda = 1/6 \) of the maximum power in the Periodogram. Any period with power greater than the threshold is considered a candidate period. We then narrow down the range of candidate periods further by utilizing the ACF. If a candidate period is not present in the ACF peak, it is discarded.

- **Step 2: Ranking.** In this step, we calculate the comprehensive ranking of candidate periods. The candidate periods near each peak of ACF are grouped together, and their ranks in the ACF and Periodogram are calculated respectively. We then calculate a score for each period based on its ranking in the periodogram and ACF. The period with the smallest score is considered as the best period in the group. We set \( \gamma = 0.6 \) to determine the score, but we will discuss the range of \( \gamma \) later.

We derive the period from the following equation:

\[ A(f) = \text{Amplitude}(\mathcal{F}(x')) \]  

(3a)

\[ P(f) = \frac{A^2(f)}{n} \]  

(3b)

\[ p_{\text{thre}} = \lambda \times \arg \max_f (P) \]  

(3c)

\[ h_{\text{peak}}(x) = \text{PeakHeight}(ACF(x)) \]  

(3d)

\[ z = \left\lfloor \frac{1}{f} \right\rfloor \]  

(3e)

\[ f_{\text{cand}} = \{ f | P(f) > p_{\text{thre}} \cap ACF(z) > \frac{1}{2}h_{\text{peak}}(z) \} \]  

(3f)

\[ p_{\text{cand}} = \left\lfloor \frac{1}{f_{\text{cand}}} \right\rfloor \]  

(3g)

\[ \text{score}(i) = \gamma \times u_i + (1 - \gamma) \times v_i, \quad i \in p_{\text{cand}} \]  

(3h)

\( p_{\text{thre}} \) is the minimum power threshold for a period to be considered a candidate. \( h_{\text{peak}} \) is the ACF peak height. \( p_{\text{cand}} \) is a period that satisfies both the Periodogram and ACF conditions. \( u_i \) and \( v_i \) are the rankings of \( i \) in the Periodogram and ACF. \( \text{score}(i) \) is the combined score of the Periodogram and ACF.
The true period is close to the periodogram and ACF peaks, despite spectral leakage and noise interference. Autoperiod and robust period methods use these peaks to find the period, but we are the first to rank candidate periods using both the ACF and periodogram. Higher rankings in both increase the probability of the true period. Extensive experiments showed that a score ranking with \( \gamma = 0.6 \) was most accurate. A larger \( \gamma \) implies a greater weight of the periodogram in the result, making the periodic detection more dependent on the ranking of the periodogram. Here, \( \gamma = 0.6 \), so the weight of the periodogram is slightly larger than that of the ACF, but they are very close.

Figure 2 shows the process of detecting periodicity in electricity demand data for England and Wales from June 5, 2000, to August 27, 2000. The data was sampled at a half-hour interval, and we hypothesized that the periods would be 48 (a day) and 336 (a week). The ACF plot revealed a peak at 24, which we disregarded. The Periodogram plot showed many green dots near 48 and 336, which we selected as candidate periods using a threshold of maximum power of \( \lambda = 1/6 \). However, due to noise in the data, it is inaccurate to rely solely on the ACF peak. Therefore, we used both the Periodogram and ACF to narrow down the candidate periods, selecting those whose power exceeded the threshold in the Periodogram and stayed near the ACF peak. In Figure 2, five points near 336 had greater power than the threshold in the Periodogram, but they were not continuous. To address this, we interpolated a cubic curve and identified the valid candidate period near 336 as ranging from 326 to 344 in the ACF. We then calculated the rankings of the candidate periods and obtained a score based on the two rankings. The period with the highest ranking (lowest score) was the best candidate period, which was 336.

**C. Quantitative Decomposition**

We utilize the period obtained through periodic detection to decompose the signal, resulting in two distinct components: trend extraction and season extraction. The decomposition process is illustrated in Figure 3.

**Step 1: Season Extraction.** The process of decomposing a multi-period time series is identical for each period. We decompose the series from high frequency (short period) to low frequency (long period) in a sequential manner. In particular, we avoid using classic filtering techniques such as Butterworth or Chebyshev filters, which require the setting of different parameters for different datasets. Incorrectly set parameters can lead to inaccurate data. Instead, we leverage the Fast Fourier Transform (FFT) to filter the data and extract seasons based on their respective periods. We obtain the season by using the following equation:

\[
 f_p = 1/\text{period} \quad (4a)
\]

\[
 HF(f) = \begin{cases} 
 1, & f \geq f_p, \\
 0, & f < f_p. 
\end{cases} \quad (4b)
\]

\[
 A(f) = \text{Amplitude}(F(x)) \quad (4c)
\]

\[
 AS(f) = A(f) \times HF(f) \quad (4d)
\]

\[
 S(x) = ||F^{-1}(AS(f))|| \quad (4e)
\]

In this equation, \( f_p \) represents the cut-off frequency, \( HF(f) \) is the function used to obtain the high frequency amplitude, \( AS(f) \) represents the amplitude of the season, and \( S(x) \) represents the season itself.

**Step 2: Extracting Trends.** After removing all of the seasonal components, the remaining parts consist of trends and residuals. To extract the trend, we apply a low-pass filter. The trend is derived from the following equation:

\[
 f_p = 1/\text{period} \quad (5a)
\]

\[
 LF(f) = \begin{cases} 
 1, & f < f_p, \\
 0, & f \geq f_p. 
\end{cases} \quad (5b)
\]

\[
 A(f) = \text{Amplitude}(F(x)) \quad (5c)
\]

\[
 AT(f) = A(f) \times LF(f) \quad (5d)
\]

\[
 T(x) = ||F^{-1}(AT(f))|| \quad (5e)
\]

The symbol \( f_p \) represents the cut-off frequency, while \( LF(f) \) refers to the function used to calculate the amplitude at low frequencies. The term \( AT(f) \) denotes the amplitude of the seasonal variation, and \( T(x) \) represents the trend.
Using SR to extract residuals from seasons with the smallest period

Fig. 3. Decomposition process: Extract Season 1, Season 2, Trend from Spectrum’s red, yellow, and green using inverse fast Fourier transform (IFFT). Extract residual with smallest period using SR.

D. Accurate Residual

In this paper, we use the Spectral Residual (SR) model to detect outlier positions and extract them. Residuals, which represent noise, are typically high-frequency and random. By leveraging the SR model, we identify residual locations in the most high-frequency seasonal terms, smooth these points, and compute the difference between the original and smoothed seasons, which yields the residual. Notably, Microsoft introduced the use of the SR model for time series analysis in 2019, marking the first instance of applying the SR model from visual saliency detection to detect anomalies in time series data.

III. EXPERIMENTS

A. Datasets

In our experiments, we utilized a combination of public and synthetic datasets, the characteristics of which are outlined in Table 1. For instance, the "M4-Hourly" dataset is a multi-period time series with a period distribution ranging from 24 to 168. This dataset consists of a total of 10 sequences, each with a sequence length of 700 data points.

To generate the synthetic data in Figure 4, we start by creating a trend signal with 700 data points. This signal includes a triangle wave to depict a gradual trend change. Then, we incorporate two cosine waves with periods of 20 and 70. Finally, we add random data for the residuals to simulate real-world situations.

TABLE I
Data Set Statistics

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Periodics</th>
<th>Periods</th>
<th>Count</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRAN</td>
<td>Non-period</td>
<td>None</td>
<td>17</td>
<td>24 – 827</td>
</tr>
<tr>
<td>CRAN</td>
<td>Single-period</td>
<td>2–52</td>
<td>58</td>
<td>24 – 3024</td>
</tr>
<tr>
<td>M4-Hourly</td>
<td>Multi-period</td>
<td>(24,168)</td>
<td>10</td>
<td>700</td>
</tr>
<tr>
<td>AusGrid-Energy</td>
<td>Multi-period</td>
<td>(48,336)</td>
<td>5</td>
<td>4416</td>
</tr>
<tr>
<td>Synthetic</td>
<td>Multi-period</td>
<td>(20,70)</td>
<td>1</td>
<td>700</td>
</tr>
</tbody>
</table>

Fig. 4. The synthetic Data

B. Setup

We have selected four state-of-the-art (SOTA) algorithms for periodicity detection comparison: findfrequency, sazed, Autoperiod, and Robust-period. Both Autoperiod and Robust-period are capable of multi-period detection and single-period detection. For the purpose of decomposition comparison, we have chosen three SOTA decomposition algorithms: MSTL, STR, and TBATS. These are all multi-period time series decomposition algorithms and can also be used for decomposing single-period time series. The details of the baseline algorithms are presented in Table II. Through a grid search on the validation set, we have set \( \lambda = 1/6 \) and \( \gamma = 0.6 \).

C. Periodic Detection and Run-Time Cost

Table III compares different period detection algorithms. ASTD performs well in detecting periods for multi-period, single-period, and non-periodic time series with the lowest false positive rate. Sazed and robust-period have lower
precision for detecting single-period time series. Autoperiod recognizes some aperiodic sequences but has a higher false positives rate. In multi-period detection, ASTD achieves the highest F1 value, while Autoperiod and Sazed detect only one cycle, and robust-period detects multiple cycles. Findfrequency obtains the period using an autoregressive model, but it is prone to interference from multi-period data. ASTD effectively mitigates the side effects of spectral leakage and abnormal values, making it the only algorithm that can correctly detect periods in time series with an outlier ratio from 0% to 20%.

Table V shows the running time comparison of different multi-period detection algorithms in different datasets. From the table, we can see that the running time of each algorithm increases with the amount of data. In the same dataset, ASTD has the shortest running time and Robust-period the longest. In the Taylor dataset, Robust-period runs 6882 times longer than ASTD. Therefore, in the case of long time series, ASTD is faster than auto-period and Robust-period, which can better extend the application of long time series.

D. Decomposition and Run-Time Cost

To quantitatively evaluate the performance, we compared the mean squared error (MSE) and mean absolute error (MAE) in the synthetic dataset, and ASTD outperformed other algorithms (Table VI).

---

### TABLE II
Source of the baseline algorithms

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>periodicity-detection</td>
<td>Findfrequency</td>
<td><a href="https://cran.r-project.org/web/packages/sazedR/index.html">https://cran.r-project.org/web/packages/sazedR/index.html</a></td>
</tr>
<tr>
<td></td>
<td>Sazed</td>
<td><a href="http://cran.r-project.org/web/packages/sazedR/index.html">http://cran.r-project.org/web/packages/sazedR/index.html</a></td>
</tr>
<tr>
<td></td>
<td>Auto-period</td>
<td><a href="https://github.com/akofke/autoperiod">https://github.com/akofke/autoperiod</a></td>
</tr>
<tr>
<td></td>
<td>Robust-period</td>
<td><a href="https://github.com/ariaghora/robust-period">https://github.com/ariaghora/robust-period</a></td>
</tr>
<tr>
<td>decomposition</td>
<td>MSTL</td>
<td><a href="https://github.com/KishManani/MSTL">https://github.com/KishManani/MSTL</a></td>
</tr>
<tr>
<td></td>
<td>STR</td>
<td><a href="https://cran.r-project.org/web/packages/stR/index.html">https://cran.r-project.org/web/packages/stR/index.html</a></td>
</tr>
<tr>
<td></td>
<td>TBATS</td>
<td><a href="https://cran.r-project.org/web/packages/tbats.html">https://cran.r-project.org/web/packages/tbats.html</a></td>
</tr>
</tbody>
</table>

### TABLE III
Detected periodicities on different dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CRAN (Non-period)</th>
<th>CRAN (Single-period)</th>
<th>M4 (Multi-period)</th>
<th>AusGrid (Multi-period)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>precision</td>
<td>recall</td>
<td>F1</td>
<td>precision</td>
</tr>
<tr>
<td>Findfrequency</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
<td>0.60</td>
</tr>
<tr>
<td>Sazed</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.65</td>
</tr>
<tr>
<td>Autoperiod</td>
<td>0.64</td>
<td>0.64</td>
<td>0.64</td>
<td>0.24</td>
</tr>
<tr>
<td>Robust-period</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.19</td>
</tr>
<tr>
<td>ASTD</td>
<td>0.76</td>
<td>0.76</td>
<td>0.76</td>
<td>0.98</td>
</tr>
</tbody>
</table>

### TABLE IV
Comparison of periodic detection for single-period time series with different outlier ratio.

<table>
<thead>
<tr>
<th>Methods</th>
<th>$\eta = 0%$</th>
<th>$\eta = 5%$</th>
<th>$\eta = 10%$</th>
<th>$\eta = 15%$</th>
<th>$\eta = 20%$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Findfrequency</td>
<td>71</td>
<td>71</td>
<td>83</td>
<td>91</td>
<td>83</td>
</tr>
<tr>
<td>Auto-period</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>67</td>
<td>67</td>
</tr>
<tr>
<td>ASTD</td>
<td>70</td>
<td>70</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
</tbody>
</table>

---

In terms of decomposition time, ASTD was found to be 3 times faster than state-of-the-art algorithms (Figure 5). Overall, ASTD performed well in recovering the location and magnitude of seasonality and residuals. Its decomposed seasonal and residuals had smaller MAE and MSE than other algorithms, and its decomposition trend was close to the current state-of-the-art algorithms. ASTD’s low time complexity makes it suitable for decomposing long time series.
Multi-period time series decomposition is a fundamental technique widely used in anomaly detection, prediction, and classification. However, the current state-of-the-art decomposition algorithms require users to identify the periodicity of time series and select different decomposition methods accordingly. In this paper, we propose a unified method called ASTD, which provides an automatic decomposition of time series, greatly expanding the scope of applications for time series decomposition. In particular, the proposed ASTD requires only the input of time series data, and can automatically detect the periodicity and decompose the sequence. We have designed ASTD with low time complexity requirements to handle long time series that are commonly encountered in real-world scenarios. The complexity of ASTD is \(O(N \log N)\). Extensive experiments have demonstrated that ASTD outperforms existing SOTA algorithms in terms of decomposition accuracy and run-time cost. Besides, it is highly robust to outliers. Furthermore, our method achieves the smallest Mean Absolute Error (MAE) and Mean Squared Error (MSE), and it is nearly 3 times faster than the state-of-the-art decomposition algorithm on the Taylor dataset. In the future, we will continue to improve the accuracy of trend extraction and explore the application of ASTD in various time series-related tasks.

V. ACKNOWLEDGEMENT

This work was supported by National Key R&D Program of China under Grant 2020YFA0711400, and National Science Foundation of China under Grant U21A20452, U19B2044.

REFERENCES


TABLE VI

<table>
<thead>
<tr>
<th>Methods</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>trend season=20</td>
<td>season=70</td>
</tr>
<tr>
<td>MSTL</td>
<td>0.2108</td>
<td>3.1543</td>
</tr>
<tr>
<td>STR</td>
<td>0.1714</td>
<td>0.2161</td>
</tr>
<tr>
<td>TBATS</td>
<td>0.8824</td>
<td>0.1981</td>
</tr>
<tr>
<td>ASTD</td>
<td>0.2463</td>
<td>0.0799</td>
</tr>
</tbody>
</table>

IV. CONCLUSION

Multi-period time series decomposition is a fundamental technique widely used in anomaly detection, prediction, and classification. However, the current state-of-the-art decomposition algorithms require users to identify the periodicity of time series and select different decomposition methods accordingly. In this paper, we propose a unified method called ASTD, which provides an automatic decomposition of time series, greatly expanding the scope of applications for time series decomposition. In particular, the proposed ASTD requires only the input of time series data, and can automatically detect the periodicity and decompose the sequence. We have designed ASTD with low time complexity requirements to handle long time series that are commonly encountered in real-world scenarios. The complexity of ASTD is \(O(N \log N)\). Extensive experiments have demonstrated that ASTD outperforms existing SOTA algorithms in terms of decomposition accuracy and run-time cost. Besides, it is highly robust to outliers. Furthermore, our method achieves the smallest Mean Absolute Error (MAE) and Mean Squared Error (MSE), and it is nearly 3 times faster than the state-of-the-art decomposition algorithm on the Taylor dataset. In the future, we will continue to improve the accuracy of trend extraction and explore the application of ASTD in various time series-related tasks.

A novel subjective bias detection method based on multi-information fusion

Lidan Zhao, Tong Li, Zhen Yang, Junrui Liu

Beijing University of Technology
litong@bjut.edu.cn

Abstract—News and social media messages usually contain subjective opinions conflicting with the needs of readers who want to receive objective information through public channels. To this end, the detection of subjectively biased sentences has become an important research issue. However, existing subjective bias detection approaches lack considering the syntactic structure and topical context of biased descriptions. In this paper, we propose a Subjective bias deTection mEthod (SITE) that comprehensively fuses multiple bias-relevant information. Specifically, we first investigate the modification and lexical features of biased sentences, based on which we formulate a set of rules to characterize biased sentences. Then, we extract the semantic features of sentences using the BERT model, based on which we further mine topic features by clustering semantically similar sentences. Finally, we comprehensively characterize biased sentences by fusing such features and train a classification model to detect biased sentences in social media. We conducted a series of experiments on a public dataset, the results of which show that SITE can detect biased sentences with 86.2\% accuracy, outperforming baseline methods.

Index Terms—Bias Detection, Dependency Structure, Topical Context

I. INTRODUCTION

In recent years, there has been an increasing amount of research on bias in textual representations, including word bias detection and sentence bias detection. Word bias can be caused by people’s stereotypes, such as gender and racial bias. The current research on sentence bias is mainly concerned with subjective bias. Subjective bias occurs when the language that should be neutral and fair is skewed by feeling, opinion, or taste (whether consciously or unconsciously) [1]. People may inadvertently add their subjective ideas when recording opinions, introducing subjective bias into the text and affecting readers’ thinking. For example, “John McCain exposed as an unprincipled politician”, which expresses the editor’s negative view of the subject, would be more neutrally expressed as “John McCain described as an unprincipled politician”. By studying bias in text, objective facts can be conveyed to readers more neutrally. In this way, the independent thinking ability of readers can be improved, and at the same time, readers can be more clear about their views on objective things.

Existing works believe that classifying subjective bias can better identify the words that trigger bias in a text. They detect biased sentences by identifying specific lexical cues [1], [2]. Some scholars focus on syntactic features to classify sentences [3]. However, extracting linguistic cues or certain syntactic features is insufficient to detect subjective bias because this information is present in most sentences. In addition, biased sentences have some other common features which help implement the bias detection task. Some scholars use deep learning models to mine the hidden semantic representation of sentences and combine sentence semantics and syntactic information as features for bias detection [4], [5]. However, these studies ignore other information, such as the topic information of sentences, bias category information, etc. The topic of the sentence can represent the background information associated with the semantics of the sentence. Sentences with similar topics have a close probability of bias, so topic features are also an important feature for detecting bias.

In this paper, we propose a Subjective Bias deTection mEthod (SITE) that comprehensively investigates and combines dependency, semantic, and topic features of sentences. Specifically, dependency features are defined by systematically investigating biased sentences and condensing biased features, resulting in a series of features of modification structure. Moreover, we mine the deep semantic features of sentences by adopting the Bidirectional Encoder Representations from Transformers (BERT) [6] model to complement the dependency features. On top of such semantic features, we further identify the topical context of sentences by clustering the sentence embeddings. Eventually, we fuse these three types of features and train a subjective bias detection model. The main contributions proposed in this paper are as follows:

- We investigate the modification structures that tend to trigger subjective bias and combine them with biased words to define dependency features.
- We consider the topical context of sentences by investigating and mining potential relationships among similar semantic sentences, based on which we propose a subjective bias detection method that combines dependency features, semantic features, and topic features.
- We conducted experiments on a public dataset, the results of which show that SITE can detect biased sentences with 86.2\% accuracy.

The rest of the paper is structured as follows: Section II discusses related works. Section III describes the design details of the proposed approach. In section IV, we evaluate our approach on a public dataset and compare the performance with the existing approaches. Finally, we conclude in section V and look forward to future works.

DOI reference number: 10.18293/SEKE2023-103.
II. RELATED WORK

Recent studies on bias focus on word bias detection [7]–[9] and sentence bias detection [3]–[5], [10].

A. Word bias detection

Researchers use models such as Word2vec [11] and GloVe [12] to embed the words as vectors, comprehensively use the relationships between words, and apply contextual semantics to detect biased words. Bolukbasi et al. [7] first identify the bias subspace and determine the direction in which the embedding vector captures the bias. They then calculate the cosine distance to judge the similarity between the embedding vectors and the bias vectors, thereby detecting the presence of bias. Kumar et al. [8] define an indirect bias based on this for studying gender bias, which considers not only the relationship between each word in the text and the bias vector but also whether the two-word vectors are strongly related. Manzini et al. [9] determine the presence of bias by calculating the similarity between the detected words and each bias vector. These studies detect word-level bias by comparing the biased word vector direction with the word vector direction.

In their work, they consider the vector relationship between words and biased words to implement the word bias detection task. When we perform sentence bias detection, similar to this, we use hidden vectors to represent various features of sentences and then detect whether bias exists.

B. Sentence bias detection

Researchers detect bias by extracting lexical, syntactic, semantic, and other features of sentences. May et al. [10] use the Sentence Encoder Association Test (SEAT) method to map sentences into fixed-size vectors and determine whether a sentence is biased or not by calculating the salience of the association and the size of the association in different vector spaces. Liang et al. [13] refer to the bias between words in a sentence as fine-grained local bias and the bias between semantics in a sentence as high-level global bias. Sentence-level bias detection is performed in two parts. The first part is calculating the contextual probability between the word vector and bias vector to identify local bias. The second part identifies global bias through sentence sentiment scores and regard scores.

Hube et al. [3] focus on cases of sentence-level linguistic bias in Wikipedia and propose DMSW. DMSW is a supervised classification method that relies on automatically creating biased words, as well as other syntactic and semantic features of biased statements. They analyze the proportion of words with bias in the sentence and their context, LIWC [14] features, and the framing bias and epistemic bias of the words in context for bias detection. Hube et al. [5] focus on the specific case of phrasing bias, which may be introduced through specific inflammatory words or phrases in a statement. They propose an RNN-based classification model for biased statements, extracting sentence hidden semantic representations to capture the inter-dependencies between words in phrases that introduce bias while incorporating LIWC features for text classification.

Pant et al. [4] explore various BERT-based models, including BERT, RoBERTa, and ALBERT, with their base and large specifications along with their native classifiers. To extract the semantic information of sentences, an integrated BERT-based model is proposed for detecting subjective bias in Wikipedia.

III. METHOD

Our method consists of three modules: dependency feature extraction module, semantic feature extraction module, and topic feature extraction module. The overview of SITE architecture is shown in Fig. 1.

The task of the first module is to extract the dependency features of sentences. According to the dependency analysis results, SITE determines and fuses the modification structure. At the same time, SITE determines the biased words and calculates the proportion of the biased words to extract the dependency vector of a sentence. In the second module, we use the BERT model to consider the correlation between tokens to generate global information about the sentence, that is, the semantic vector of the sentence. The third module extracts the topic features of sentences. SITE trains a topic model by clustering the embedding vectors of sentences and extracting topic words for each cluster using a class-based variant of TF-IDF. Finally, the semantic vector of the sentence is used as the input of the model to obtain the topic vector. The three feature vectors are concatenated as the feature vector of the sentence. Based on these features, the classifier identifies whether there is a subjective bias in the sentence.

A. Dependency feature extraction

By investigating the common features of biased sentences, we find that some sentence structures or words are less likely to appear in neutral sentences. SITE defines these modification structures and lexical features in these sentences as dependency features.

1) Modification features: The longer a sentence is, the more complex its syntactic structure is, and the richer its meaning is. Biased sentences often convey subjective ideas through the use of nested modification structures. Such modification structures are constructed based on the words in the sentence. Different relationships between words will be biased to varying degrees. In biased sentences, the relationship that has a large influence on subjective bias appears more likely. Therefore, we perform dependency analysis on the sentences and then define the modification structures associated with bias.

Dependency parsing shows the subordination and modification relationship between words. The dependencies of a sentence can be represented by a diagram.

For example: “the British Broadcasting Corporation or BBC is the most widely respected broadcasting organisation in the world”. The analysis of the dependencies is shown in Fig. 2. There is an arrow pointing from “organisation” to “broadcasting”, indicating that “broadcasting” modifies “organisation”. The relationship on the arrow is “compound”, indicating that “broadcasting” is a noun compound of “organisation”, which is used to modify “organisation”. Similarly, “respected”
depends on “organisation”, which is an adjective modifier of “organisation”. Furthermore, “det” is the relation between the head of an NP and its determiner. “nsubj” is a nominal which is the syntactic subject and the proto-agent of a clause. “cc” is the relation between an element of a conjunct and the coordinating conjunction word of the conjunct. “advmod” is a (non-clausal) adverb or adverbial phrase (ADVP) that serves to modify the meaning of the word. In addition to these relationships, the other relationships represent the modification relationship between the words at both ends of the arrow. For specific meanings, please refer to the Universal Dependencies website for more information. In Fig. 2, the tags below the words are part-of-speech (POS) tags of the words. “DT” stands for determiner. “NNP” stands for proper noun. “CC” stands for coordinating conjunction. “RBZ” stands for 3rd person singular. “RBS” stands for the superlative. In addition to these POS tags, more detailed explanations of the labels can be found on the Universal Dependencies website.

Based on the dependency parsing of biased sentences, we find some common modification structures in biased sentences that introduce optimistic or pessimistic views of objective facts. The modification structures we define include the same noun or proper noun modified by multiple adjectives, and a noun or proper noun modified by both an adjective and a noun. For example, in Fig. 2, “the most widely respected broadcasting organisation” is an explanation of the BBC. In the sentence, “organisation” is modified two times to express a stronger admiration, which leads to a biased sentence. In such a case, the likelihood of adding subjective emotions is greatly increased. In addition to this, we extract several other structures (Table I). We define such structures as atomic features (AF), modification structures that are more likely to occur in biased sentences.

Complex modification structures help editors express their subjective emotions, so sentences with multiple modification features are more likely to develop a subjective bias (Table I). For example, the structure AF1 and the structure AF4 are more likely to coexist in biased sentences. Therefore, we determine whether both structure AF1 and structure AF4 are present in the sentence. In addition, we combined structures AF2 and AF5, and structures AF1, AF3, and AF4. The details of the combined features are shown in Table II.
TABLE I
DESCRIPTION OF THE MODIFICATION STRUCTURE.

<table>
<thead>
<tr>
<th>No.</th>
<th>Structure</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF1</td>
<td><img src="image1" alt="AF1 Diagram" /></td>
<td>“the Bugatti Veyron 16.4 is the fastest (dependent1) street-legal (dependent2) production car (head) in the world.”</td>
</tr>
<tr>
<td>AF2</td>
<td><img src="image2" alt="AF2 Diagram" /></td>
<td>“brilliant (dependent1) but pompous (dependent2) entrepreneur Mark (head) is the envy of all his colleagues.”</td>
</tr>
<tr>
<td>AF3</td>
<td><img src="image3" alt="AF3 Diagram" /></td>
<td>“The Great Global Warming Swindle argues against prominent (dependent1) scientific (dependent2) views (head) on global warming.”</td>
</tr>
<tr>
<td>AF4</td>
<td><img src="image4" alt="AF4 Diagram" /></td>
<td>“African poverty is due to the rampant (dependent1) government (dependent2) corruption (head) on that continent.”</td>
</tr>
<tr>
<td>AF5</td>
<td><img src="image5" alt="AF5 Diagram" /></td>
<td>“famous (dependent1) Indian singer (dependent2) Sonu Nigam (head) sang many songs of Akhlaq Ahmed.”</td>
</tr>
<tr>
<td>AF6</td>
<td><img src="image6" alt="AF6 Diagram" /></td>
<td>“his novels were real (dependent1) page-turners (dependent2-head), but grounded on meticulous historical research.”</td>
</tr>
</tbody>
</table>

TABLE II
FUENDING THE MODIFICATION STRUCTURE.

<table>
<thead>
<tr>
<th>No.</th>
<th>Structure</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF1</td>
<td>AF1&amp;AF4</td>
<td>“Missouri governor Lilburn Boggs issued the ominous sounding extermination order (AF1&amp;AF4).”</td>
</tr>
<tr>
<td>CF2</td>
<td>AF2&amp;AF5</td>
<td>“Kristin Shepard is a fictional character on the popular American television series (AF2&amp;AF5).”</td>
</tr>
<tr>
<td>CF3</td>
<td>AF1&amp;AF3&amp;AF4</td>
<td>“the most famous German “Panzer Ace” (AF1&amp;AF4) is credited by Kurowksi as having destroyed 60 tanks and nearly as many anti-tank guns (AF3).”</td>
</tr>
</tbody>
</table>

2) Subjective bias lexical features: Dependency parsing is a more detailed analysis of the syntax in a sentence. It analyses the relationship between the constituents of a sentence, with less analysis of the semantic aspects. For example, in the analysis of “the most widely respected broadcasting organisation” and “the most widely regarded broadcasting organisation”, the dependency parsing can be interpreted as a modification of the organisation. However, in the actual context, the former expresses a positive sentiment towards the organisation. This shows that the identification of the modification features in the sentence alone does not reveal the features that trigger the bias. Therefore, it is necessary to pay attention to the biased words in the sentences, especially in the modification structures.

Subjective bias lexical features mainly refer to biased words. Biased words mainly capture the subjective sentiment expressed in the sentence, including factive verbs, assertive verbs, implicatives, and other entailments, hedges, and subjective intensifiers [2]. For example, “say” and “state” are usually neutral, “pointing out” and “claim” cast doubt on the certainty of the proposition [2].

Suppose the modification structure in Table I, II exists in a sentence, and the subordinate words in the structure are words with subjective sentiment. In this case, the sentence is more likely to have subjective bias. In this paper, we use the biased words table provided by Pryzant et al. [1] to identify bias modifiers in modification structures.

A biased word is the smallest unit of biased expression in a sentence, and it is a way to express subjective ideas. Therefore, in addition to incorporating it into the modification structure, we consider the number of biased words in the sentence. In a sentence, if there are more biased words, there is a greater probability that the sentence has a subjective bias. Therefore, it makes sense to use the proportion of biased words in a sentence as auxiliary information for the bias detection task.

Our method extracts the features defined above as dependency vectors of sentences, which are used in the bias detection task.

B. Semantic feature extraction

The representation of subjective bias in sentences is subtle, and merely extracting dependency features is insufficient to detect bias. In addition to this, semantics is crucial information for sentence classification tasks. BERT [6] is an Autoencoder language model that works well with contextual information to generate sentence semantic vectors. It uses the Masked Language Model to pre-train a bidirectional Transformer to generate language representations capable of incorporating contextual information. It is trained on a large corpus, and then the model is fine-tuned for our tasks by adding some extra layers at the end, which can be classification, question answering.

In this paper, we fine-tune the BERT model [6] to extract hidden vector representations of sentences. The input of BERT is the representation of each token in the sentence. To complete the classification task, in addition to the token, a specific classification token ([CLS]) needs to be inserted at the beginning of the sequence. BERT focuses on information from different representation subspaces at different positions by using the multi-head attention mechanism of the encoder in the Transformer and weighs the correlation between words. Specifically, the input is represented by \((x_1, x_2, ..., x_n)\), and the corresponding embedding vector \((a_1, a_2, ..., a_n)\) is generated through the BERT embedding layer. Multiple heads are generated by computing the embedding vector with multiple sets of \(Q, K, \text{ and } V\) using an attention mechanism. Then, it merges the heads and dot-multiplies \(W^O\) for a linear
transformation to generate an output corresponding to each token. The specific calculation is as (1):

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_n)W^O$$

where $head_i = Attention(QW^Q_i, KW^K_i, VW^V_i)$

(1)

where the parameter matrices $W^Q_i \in R^{d_{model} \times d_k}$, $W^K_i \in R^{d_{model} \times d_k}$, $W^V_i \in R^{d_{model} \times d_v}$ and $W^O \in R^{d_v \times d_{model}}$. $Q$, $K$, and $V$ represent query, key, and value vectors, respectively. $d$ represents the dimension.

Therefore, after the 12 layers of encoder, the output of last layer corresponding to token is used to aggregate the representation information of the entire sequence. Because each word token has its meaning, its semantics account for a large proportion of the final semantic vector. However, [CLS] has no semantics. The output of the last layer corresponding to it can more fairly express the semantic information of the entire sentence. Therefore, by training the BERT model, we extract the output of the last transformer layer corresponding to [CLS] as the sentence semantic vector. We use this vector as part of the feature vector of the classifier for the subjective bias detection task.

C. Topic feature extraction

Subjective bias may come from differences in social background, cultural background, or other factors. The likelihood of a subjective bias occurring in a sentence varies under different topics. For example, subjective bias is more likely to occur under topics such as religion, society, competition, and politics, while it is less likely to occur under topics such as philosophy and music. Understanding the topic context can help us better understand the text and detect subjective bias in the text. For example, suppose we are analyzing a text about politics. In that case, we can better understand the ideas and political position to which the text belongs, the history of politics. In that case, we can better understand the ideas and political position to which the text belongs, the history of politics.

D. Subjective Bias Classification

After extracting dependency, semantic, and topic features, we concatenate them to represent the sentence’s feature vector. In addition, each sentence includes a label indicating whether it contains bias. We use an SVM (Support Vector Machine) [21] classification model to perform subjective bias detection.

The feature vectors we extracted have the characteristics of high dimensionality and nonlinearity separability. SVM represents the training data as points in space and constructs hyperplanes in high-dimensional or infinite-dimensional space to separate these points. This method can effectively handle high-dimensional and nonlinear data and has good interpretability. Therefore, we use SVM for classification.

SVM can use different kernel functions to classify different types of text data. By finding an optimal separating hyperplane to divide the data into two categories, SVM maximizes the model’s predictive accuracy without overfitting the training data. When predicting whether a new sentence contains bias, its feature vector is mapped to the same space, and the category to which it belongs is predicted based on which side of the margin it falls on.
IV. EXPERIMENT

In this section, we first describe our dataset and further describe the experimental setup and results.

A. Dataset

There is only one open-source dataset for subjective bias detection work. The dataset we used is the Wiki Neutrality Corpus (WNC) dataset open-sourced by Reid Pryzant et al. [1]. The data is derived from the editing history of Wikipedia. Wikipedia follows three main principles when verifying entries: neutral point of view (NPOV), available for verification, and non-original research. NPOV refers to editors presenting facts in a neutral way and recording opinions without taking a position. This dataset is widely used in work on subjective bias detection, and the NPOV principle in it fits well with our point of view. Therefore, we conducted experiments with the dataset provided by Reid Pryzant et al. [1].

The dataset is an edited data crawl on Wikipedia. It contains about 180,000 biased sentences and 360,000 neutral sentences. We randomly shuffle these sentences and split the dataset into two parts: training and test sets. The training data is then used to train the classification model and the retained test data is executed for evaluation.

B. Experimental Settings

We train a classifier on the feature set mentioned in section III. We first extract the dependency vectors of sentences according to the defined dependency features. Then we train the BERT model with the sentences and their labels as input and extract the last layer vector corresponding to [CLS] as semantic vectors. For the BERT, we use a learning rate of $2 \times 10^{-5}$, a maximum sequence length of 128, a batch size of 32, and training epochs of 3 while fine-tuning the model. Finally, we use the trained BERT model to generate sentence embedding vectors to train the BERTopic model and extract the topic probabilities of sentences as topic vectors. For BERTopic, we train the model with the following hyperparameters: top_n_words of 10, n_gram_range of (1, 1), and min_topic_size of 10. The above three feature vectors are concatenated to generate the final feature vectors. We train the Support Vector Machine (SVM) model with this vector as the input to the classifier. SVM maps the feature vectors to some points in embedding space and finds a hyperplane to segment the samples to complete the classification task.

We use precision (P), recall (R), F1 score, and accuracy (ACC) as evaluation metrics to assess the results of the experiments. Precision measures the percentage of sentences predicted as having a subjective bias that has bias. Recall measures the percentage of all biased sentences in which the bias was correctly identified. Accuracy measures the percentage of the number of predicted labels that are correct. The F1-score is the summed average of the precision and recall. Thus, better classification models have higher P, R, F1, and ACC.

We compare our method to four existing baselines that focus on the same task as ours and have been more effective over the past five years.

- Recasens et al. [2]: the method is to detect subjective bias in sentences using common linguistic cues from different bias categories.
- DBWS [3]: a supervised text classifier based on biased vocabulary and other features. This method uses a random forest classifier for the bias detection task.
- Hube et al. [5]: an approach that relies on recurrent neural networks. The method mines the inter-dependencies between words in phrases that introduce biases.
- Pant et al. [4]: BERT-based model conducts comprehensive experiments to detect subjective bias, with the same purpose of experiments in this paper.

C. Experimental Results

Table III shows the experimental results of SITE compared to baseline. The best results and the second-best results are highlighted in bold and underlined, respectively. SITE outperformed other baselines, achieving a precision of 0.829, recall of 0.713, F1 score of 0.767, and accuracy of 0.862, all of which are higher than other methods.

| TABLE III COMPARISON OF EXPERIMENTAL RESULTS. |
|---------|---|---|---|---|
|         | P  | R  | F1 | ACC |
| Recasens et al. | 0.758 | 0.206 | 0.324 | 0.712 |
| DBWS    | 0.778 | 0.276 | 0.408 | 0.732 |
| Hube et al. | 0.531 | 0.548 | 0.540 | 0.710 |
| Pant et al. | 0.733 | 0.677 | 0.704 | 0.716 |
| SITE    | 0.829 | 0.713 | 0.767 | 0.862 |

The first baseline [2] and the second baseline [3] mainly analyze the lexical and syntax of sentences. From the experimental results, it can be seen that by analyzing the lexical and syntactic features, only a small part of biased sentences can be identified. So the recall is lower, 0.206 and 0.276, respectively.

Hube et al. [5] mines the hidden semantic representation and LIWC features of sentences, etc. This method uses RNN for semantic representation. RNN cannot support long-term sequences and is not suitable for processing long sentences. But sentences with subjective bias generally have complex syntactic features and are relatively long, so the classification effect is relatively poor. Compared to Hube et al. [5], our method improved P by 29.8%, R by 16.5%, F1 by 22.7%, and ACC by 15.2%.

Pant et al. [4] using the BERT model to mine sentence information, its context can be taken into account, and the classification effect is relatively good. The overall performance of this method is second only to our method. However, the sentence information extracted by this method is single. On this basis, this paper extracts topic features, mines sentence modification features, biased words, etc., and achieves better results. Compared to the Pant et al. [4], our method improved P by 9.6%, R by 3.6%, F1 by 6.3%, and ACC by 14.6%.

This experimental result proves that considering sentence dependency, semantic, and topic features, it is possible to fully mine sentence information and improve the overall performance of the classification model.

V. CONCLUSION

In this paper, we propose a subjective bias detection model that fuses multiple kinds of information. The model fuses dependency features, semantic features, and topic features to classify sentences.

Dependency features are more inclined to judge simple sentences and sentences with explicit expressions. This is because dependency features represent modification structures in sentences. When the biased sentence is simple, modifiers to the main body of the sentence are often added to the sentence to introduce bias. When a sentence is clearly expressed, there will be a significant number of modification structures. In this case, the dependency feature of the sentence plays an important role in detecting bias, and the accuracy of the result is high. Topic features are more inclined to judge sentences with rich context. Because extracting the topic features of a sentence is equivalent to extracting various information such as background and history under a certain topic to which the sentence belongs. When there are more sentences under the same topic, the more information it contains, and the more accurate the subjective bias detection under the topic is. Semantic features are the key information for sentence feature extraction, which can handle the context information of sentences well and are applicable to any sentence. The results show that considering these three types of features can better highlight the biased features of sentences, which helps improve model performance. The completion of this classification task dramatically helps readers to read texts and also helps to detect biased texts on Wikipedia.

Subjective bias does not exist explicitly in the text, and we need to mine the hidden information. We need to interpret and extract sentence features to detect bias further. However, a complete elaboration of sentence features is complex and requires continuous research to detect bias further. It makes sense to automate the removal of subjective biases from text and generate more objective expressions. In future studies, we will conduct bias neutralization studies.

ACKNOWLEDGEMENT

This work is partially supported by the National Key R&D Program of China (No.2022YFB3103100), the Major Research Plan of the National Natural Science Foundation of China (92167102), the Project of Beijing Municipal Education Commission (No.KM202110005025), the Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions (CIT&TD20190308), and Engineering Research Center of Intelligent Perception and Autonomous Control, Ministry of Education.

REFERENCES

ASMix: An Attention-based Smooth Data Augmentation Approach

Rongxi Guo, Wanrong Jiang, Xi Zhao, Ding Cao, Guiquan Liu∗
University of Science and Technology of China, Hefei, China
{guorongxi, jwr, xixilili, caoding}@mail.ustc.edu.cn, gqliu@ustc.edu.cn

Abstract—Data augmentation through linearly interpolating inputs and modeling targets of random samples has significantly improved predictive performance. However, data augmentation based on linear interpolation generates semantically cluttered and ambiguous text, resulting in ineffective augmentation. To address these issues, in this paper, we propose a novel data augmentation approach called Attention-based Smooth Data Augmentation (ASMix). ASMix accepts the smoothed embeddings of pairwise data predicted by a masked language model (MLM) instead of one-hot embeddings, which makes the inputs more informative and context-rich. We employ the attention mechanism to select discriminative and more-attentioned parts of text hidden representations and mix up the parts containing key semantics in the hidden representations of pairwise data through a multi-token replacement strategy to augment the data of the minority class, which greatly reduces redundant information in the representations that hurts the performance of the model. On several public imbalanced text classification benchmarks, ASMix outperforms state-of-the-art data augmentation methods. In minority classes, the performance improvement of ASMix is particularly prominent.

Keywords—Data imbalance, Data augmentation, Text classification

I. INTRODUCTION

One of the challenges of machine learning is the lack of sufficient data, which leads to data imbalance problems. In the imbalanced binary classification task, the class with the larger size is known as the majority class, and the other class is known as the minority class [1].

In the case of imbalanced data, traditional classifiers learn biased models that tend to be biased towards the majority class and overfit the minority class. Many methods have been proposed. Resampling [2] has received extensive attention for its simplicity and effectiveness, including undersampling and oversampling. As an improved oversampling method, data augmentation addresses the lack of data by increasing the amount of training data. Mixup [3] mixes two samples by interpolating their images and labels to generate a virtual sample as training data in Computer Vision (CV). CutMix [5] replaces image regions with a patch of another training sample as training data in Computer Vision (CV). CutMix addresses the lack of data by increasing the amount of training data. Mixup [3] mixes two samples by interpolating their images and labels to generate a virtual sample as training data in Computer Vision (CV). CutMix [5] replaces image regions with a patch of another training sample as training data in Computer Vision (CV).

Due to the discreteness of the text input space and the complexity of the text structure, it is challenging for CutMix to apply in the text input space. If we apply it to the hidden layer, there is still a problem. Many tokens in the text are irrelevant to classification, such as stop words with no actual meaning, meaningless filler words in padding to fill the sentence length to the maximum length, and words containing other information unrelated to classification. These tokens are unimportant for text classification tasks and may also hurt the performance of classifiers. If we randomly replace the hidden representations of tokens, these stop words, filler words, and tokens unrelated to classification may be incorporated into synthetic samples, making samples incredible.

To overcome the above shortcomings, in this work, we propose a novel augmentation method called ASMix. ASMix accepts the smoothed embeddings of pair-wise data predicted by MLM instead of one-hot embeddings as model inputs, which are more informative and context-rich. [6] In the hidden space, to avoid meaningless or irrelevant information, ASMix uses the self-attention mechanism [7] to select the text information that the model pays more attention to through a multi-token replacement strategy. ASMix fully mixes the semantics related to classification in the pairwise data and generates new samples that are not similar to the original samples, which makes the decision boundary smoother and further away from the training data.

Our main contributions in this article are as follows:

- We propose a novel data augmentation method that employs a self-attention mechanism to combine pairwise smoothed samples through a local replacement strategy to generate augmented samples.
- To the best of our knowledge, we are the first to mix label-related tokens of sentences in the hidden layer to enhance the text.
- Our proposed ASMix achieves state-of-the-art performance on three imbalanced classification datasets, is particularly helpful for the minority class, and shows robustness across languages.

II. RELATED WORK

A. Imbalanced Learning

The current methods of imbalanced learning are divided into two categories: algorithm-level methods and data-level methods. The cost-sensitive method [8] is an algorithm-level approach that takes the costs into account, which improves the classifiers by assigning different costs to classes. In addition, Focal Loss [9] alleviates the class imbalance problem by...
Fig. 1: The overall architecture of ASMix.

B. Data Augmentation for Text

Data augmentation was initially widely used in CV and has since been extended to Natural Language Processing (NLP) by many researchers. Back-translation [11] generates different data on the premise of keeping the semantics of the original sentence unchanged. Guo et al. [12] first introduced Mixup [3] to the NLP task. TMix [4] synthesizes a large amount of augmented training data by interpolating in Euclidean space, while HYPMIX [13] performs interpolation operations in hyperbolic space to better capture the complex geometry of hidden state hierarchies. Previous data augmentation methods expanded all categories. Our proposed ASMix not only expands all data but also mainly augments the minority class, which can better address the data imbalance problem.

III. PROPOSED METHOD

A. Notations

Given a text dataset, $D = \{(x_i, y_i)\}_{i=1}^N$, where $N$ is the number of instances, $x_i$ is the discrete sequence of text in the input space, $y_i$ is the label of the instance, and the number of classes in dataset $D$ is $n$. $(x_A, x_B, y_A, y_B)$ represents a pair of data extracted from the dataset $D$, where $A \neq B$.

B. Representation Augmentation

BERT [14] alleviates the unidirectionality constraint by proposing a "masked language model" pre-training objective. The masked language model masks some percentage of the input tokens at random, and the objective is to predict the original vocabulary of the masked word based on its context.

Given an instance $(x_A, y_A)$, we first convert the original text $x_A$ into an one-hot encoding $\hat{x}_A$ by tokenizer. We use the one-hot encoding $\hat{x}_A$ as the input of BERT and obtain the output of the last layer of the transformer encoder in BERT, which is denoted as:

$$\hat{x}_A = \text{convert to onehot}(\text{tokenizer}(x_A))$$ (1)

$$x_A = \text{BERT}(\hat{x}_A)$$ (2)

where $x_A \in \mathbb{R}^{\text{seq-len} \times \text{emb-size}}$ is a 2D dense vector. We multiply $x_A$ and the word embedding matrix $W \in \mathbb{R}^{\text{vocab-size} \times \text{emb-size}}$ in BERT to get the prediction result of MLM, which is mainly distributed over the context-compatible tokens at this position. To address the model's preference for tokens that appear in similar contexts but conflict with task labels, we employ the interpolation, which is defined as:

$$\text{MLM}(x_A) = \text{softmax}(\vec{x}_A W^T)$$ (3)

$$\tilde{x}_A = \mu x_A + (1 - \mu) \text{MLM}(x_A)$$ (4)

where $\tilde{x}_A$ is the interpolated representation, and $\mu$ is the balance hyperparameter that controls the interpolation strength. We use $\tilde{x}_A$ as the input to the classifier instead of the one-hot representation. The one-hot representation $x_A$ and smoothed representation $\text{MLM}(x_A)$ are derived from the same raw input $x_A$. We keep the label $y_A$ unchanged.

C. ASMix

Given a pair of enhanced representation inputs $(\tilde{x}_A, \tilde{x}_B, y_A, y_B)$, we compute the hidden representations $h_A \in \mathbb{R}^{l \times d}$ and $h_B \in \mathbb{R}^{l \times d}$ of the inputs $x_A$ and $x_B$ separately in the bottom layers of the model, where $l$ is the
Similar to CutMix, we define the combining operation as:

$$\tilde{h} = (1 - M) \odot h_A + M \odot h_B$$

$$\tilde{y} = (1 - \lambda)y_A + \lambda y_B$$

(5)

where $M \in \{0, 1\}^{1 \times d}$ denotes a binary mask indicating where to drop out and fill in from two samples introduced in the following section. $\lambda$ represents the interpolation strength of the one-hot representation of the labels, and $\odot$ is element-wise multiplication. The interpolation strength $\lambda$ is calculated using the number of tokens participating in ASMix, which is consistent with the combination ratio of the representations.

D. Multi-token Replacement Strategy

We use the scaled dot-product attention [7] to explore the dependency between the target and candidates from the hidden state $h$ and compute the total attention score for each token in a sequence as follows:

$$A = h(\text{softmax}((W_q h)^T (W_k h)))$$

$$A_i^* = \sum_{j=1}^{d} A_{ij}$$

(6)

(7)

where $W_q$ and $W_k$ are trainable weights, $A \in \mathbb{R}^{1 \times d}$ is the output of the attention layer, and $A_i^*$ represents the attention score of the $i$-th token, $i \in [1, l]$. We sample the binary mask $M$ as follows:

$$M_i = \begin{cases} 
1^{1 \times d} & \text{if } A_i^* \geq \tau \\
0^{1 \times d} & \text{otherwise} 
\end{cases}$$

$$M = \text{Concat}(M_1, M_2, \cdots, M_l)$$

(8)

where Concat(.) is the concatenate operation, $\tau$ is the attention score threshold. We employ a random sampling strategy to obtain the threshold $\tau$ as follows:

$$\tau = \theta \max_i (A_i^*)$$

$$\theta \sim \text{Beta}(\alpha, \alpha)$$

(9)

Here, the parameter $\theta$ is a random value sampled from a Beta distribution, and $\alpha$ is a hyper-parameter that controls the distribution of $\theta$. The threshold $\tau$ determines which tokens in the hidden representation will be combined in ASMix.

E. Framework of Text Classification

As shown in Fig. 1, we implement ASMix at one of the layers of BERT. For an imbalanced text dataset $D$, $n$ classes are represented as $\{C^0, \ldots, C^{n-1}\}$, where $C^0$ is the majority class with the most samples. For the other class $C^i$, $i \in [1, n-1]$, the Imbalance Ratio (IR) of the class is the ratio of $\text{size}(C^0)$ and $\text{size}(C^i)$.

For a class $C^i$, $i \in [1, n-1]$, we determine the sampling ratio $r = |\text{IR} - 1|$. For a sample $x$ in the class $C^i$, we randomly sample $r$ samples $\{x_1^{aug}, \ldots, x_r^{aug}\}$ in the training set $D$, resulting in $r$ sample pairs $\{(x, x_1^{aug}), \ldots, (x, x_r^{aug})\}$. For the majority class $C^0$, there is no need to sample new samples, $x^{aug} = x$.

We use the interpolation representation $(\tilde{x}, \tilde{z}^{aug})$ of a text pair $(x, z^{aug})$ as the input of BERT, which is transformed into a pair of hidden representations $(h, h^{aug})$. As an encoder, BERT has $L$ layers. We choose to employ ASMix at the $m$-th layer, where $m \in [0, L]$. The $l$-th layer in the network is represented as $f_l(\cdot; \theta)$. The hidden representation of layer $l$ can be computed with $h_l = f_l(h_{l-1}; \theta)$. First, we compute the hidden representations of the two text samples separately in the bottom layers of the model:

$$h_l = f_l(h_{l-1}; \theta), l \in [1, m]$$

$$h_l^{aug} = f_l(h_{l-1}^{aug}; \theta), l \in [1, m]$$

(10)

Then, we execute ASMix at the $m$-th layer and continue to pass the augmented representation to the upper layers of

### TABLE I: Dataset statistics and dataset split.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Class</th>
<th>Train</th>
<th>Test</th>
<th>IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>R8</td>
<td>earn</td>
<td>2675</td>
<td>1040</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>acq</td>
<td>1438</td>
<td>637</td>
<td>1.859</td>
</tr>
<tr>
<td></td>
<td>trade</td>
<td>230</td>
<td>64</td>
<td>11.622</td>
</tr>
<tr>
<td></td>
<td>crude</td>
<td>222</td>
<td>113</td>
<td>12.041</td>
</tr>
<tr>
<td></td>
<td>money-fx</td>
<td>168</td>
<td>69</td>
<td>15.911</td>
</tr>
<tr>
<td></td>
<td>interest</td>
<td>144</td>
<td>66</td>
<td>18.563</td>
</tr>
<tr>
<td></td>
<td>ship</td>
<td>107</td>
<td>35</td>
<td>24.981</td>
</tr>
<tr>
<td></td>
<td>grain</td>
<td>37</td>
<td>10</td>
<td>72.243</td>
</tr>
<tr>
<td>Cade12</td>
<td>serviços</td>
<td>5627</td>
<td>2346</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>sociedade</td>
<td>4935</td>
<td>2428</td>
<td>1.140</td>
</tr>
<tr>
<td></td>
<td>lazer</td>
<td>3698</td>
<td>1892</td>
<td>1.522</td>
</tr>
<tr>
<td></td>
<td>internet</td>
<td>1585</td>
<td>796</td>
<td>3.550</td>
</tr>
<tr>
<td></td>
<td>noticias</td>
<td>701</td>
<td>381</td>
<td>8.027</td>
</tr>
<tr>
<td></td>
<td>compras-online</td>
<td>423</td>
<td>202</td>
<td>13.303</td>
</tr>
<tr>
<td>THS</td>
<td>0</td>
<td>20793</td>
<td>8914</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1571</td>
<td>670</td>
<td>13.236</td>
</tr>
</tbody>
</table>
the model, the \((m+1)\)-th to \(L\)-th layers. The labels are also interpolated with the strength consistent with the combination ratio of the hidden states. The definition is as follows:

\[
\hat{h}_m = (1 - M) \odot h_m + M \odot h_m^{aug}
\]

\[
\hat{h}_l = f_l(\hat{h}_{l-1}; \theta), l \in [m+1, L]
\]

In text classification, we implement the classifier as a two-layer MLP following BERT. It takes the representation of the BERT output as input and returns a probability vector. We train the entire model by minimizing the Cross-entropy between the interpolated labels and the probability from the classifier as follows:

\[
L = - \sum \gamma \log \text{ASMix}(\tilde{x}_\gamma, \tilde{x}_\gamma^{aug}) + (1 - \gamma) \log (1 - \text{ASMix}(\tilde{x}_\gamma, \tilde{x}_\gamma^{aug}))
\]

where \(\text{ASMix}(\cdot, \cdot)\) represents the output of the classifier.

### IV. Experiments

#### A. Datasets

We conducted experiments on three benchmark imbalanced text datasets.

- **R8** [13] is a corpus of Reuters news articles in English containing 29,930 words.
- **Cade12** [15] is a corpus of Brazilian web pages classified by human experts extracted from the CADÊ Web Directory.
- **THS** [16] is an imbalanced binary dataset for Twitter hate speech detection. The label ‘1’ denotes the tweet is racist or sexist, and the label ‘0’ represents the tweet is neither racist nor sexist.

We used the original training and testing sets for the first two datasets as our training and testing sets. For the last dataset, we split the dataset into training and test sets at a ratio of 7:3. The dataset statistics and split information are presented in Table I.

#### B. Baselines

To test the effectiveness of our method, we compared it with several recent methods. The following methods all used BERT multilingual base model as the backbone network:

- **BERT** [14]: We used a pre-trained BERT-base-multilingual-cased model and fine-tuned it for classification.
- **EDA** [17]: Easy Data Augmentation (EDA) chooses and performs one of the following operations at random for each sentence in the training set: Synonym Replacement (SR), Random Insertion (RI), Random Swap (RS), and Random Deletion (RD).
- **AEDA** [18]: An Easier Data Augmentation (AEDA) randomly inserts punctuation marks into the original text.
- **TMix** [4]: TMix takes two text examples and interpolates them in their corresponding hidden space.
- **HYPMIX** [13]: HYPMIX is an interpolative data augmentation technique operating in the hyperbolic space.

#### C. Experimental Settings

We used BERT-base-multilingual-cased tokenizer to tokenize the text, BERT-base-multilingual-cased model as our text encoder, average pooling on the output of the encoder, and a two-layer MLP with a 128 hidden size and \(\tanh\) as its activation function. We used AdamW as the optimizer and the weight decay is 0.01. The learning rates for BERT encoder and MLP are set to 1e-5 and 1e-3, respectively. We train all models for 50 epochs and set the batch size to 16. The number of heads of the multi-head attention mechanism, \(t\), is set to 8. For R8, Cade12, and THS, the maximum sentence length is 128, 150, and 50, respectively. In addition, we set a fixed seed when training the model to ensure the reproducibility of the results.

#### D. Overall Results

We used Precision (%), Recall (%), and macro-F1 (%) as metrics to evaluate our method ASMix by comparing with all the baselines mentioned above on different imbalanced text classification datasets.

The overall results on different imbalanced text classification datasets are shown in Table II. Firstly, basically all models using data augmentation outperform BERT on the F1 metric. Secondly, our method outperforms EDA, AEDA, TMix, HYPMIX in all metrics. Compared with other interpolation-based methods, TMix and HYPMIX, ASMix avoids meaningless information, resulting in a sample that more thoroughly combines the semantics of the two original instances.

#### E. Analysis of Each Category

To show the effect of our method on enhancing the minority class in the imbalanced text classification dataset, Table II.
TABLE III: Performance (F1 (%)) of each category on R8 in comparison with baselines.

<table>
<thead>
<tr>
<th>Model</th>
<th>earn</th>
<th>acq</th>
<th>trade</th>
<th>crude</th>
<th>money-fx</th>
<th>interest</th>
<th>ship</th>
<th>grain</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>98.25</td>
<td>97.29</td>
<td>89.76</td>
<td>92.45</td>
<td>89.36</td>
<td>92.42</td>
<td>92.54</td>
<td>100</td>
</tr>
<tr>
<td>EDA</td>
<td>97.96</td>
<td>97.94</td>
<td>88.36</td>
<td>92.97</td>
<td>91.17</td>
<td>95.28</td>
<td>93.94</td>
<td>100</td>
</tr>
<tr>
<td>AEDA</td>
<td>98.70</td>
<td>98.12</td>
<td>92.91</td>
<td>91.51</td>
<td>88.89</td>
<td>93.85</td>
<td>94.12</td>
<td>100</td>
</tr>
<tr>
<td>TMix</td>
<td>98.81</td>
<td>98.02</td>
<td>92.42</td>
<td>93.09</td>
<td>92.09</td>
<td>91.04</td>
<td>94.12</td>
<td>100</td>
</tr>
<tr>
<td>HYPMIX</td>
<td>98.76</td>
<td>97.72</td>
<td>91.04</td>
<td>91.00</td>
<td>92.75</td>
<td>91.73</td>
<td>95.52</td>
<td>100</td>
</tr>
<tr>
<td>ASMix</td>
<td>98.52</td>
<td>97.15</td>
<td>90.08</td>
<td>94.51</td>
<td>92.70</td>
<td>93.91</td>
<td>98.55</td>
<td>100</td>
</tr>
</tbody>
</table>

TABLE IV: Performance (F1 (%)) of each category on Cade12 in comparison with baselines.

<table>
<thead>
<tr>
<th>Model</th>
<th>servicos</th>
<th>sociedade</th>
<th>lazer</th>
<th>internet</th>
<th>noticias</th>
<th>compras-online</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>73.14</td>
<td>67.23</td>
<td>66.10</td>
<td>56.56</td>
<td>43.67</td>
<td>45.76</td>
</tr>
<tr>
<td>EDA</td>
<td>68.89</td>
<td>66.58</td>
<td>66.56</td>
<td>54.51</td>
<td>52.65</td>
<td>47.68</td>
</tr>
<tr>
<td>AEDA</td>
<td>72.00</td>
<td>64.36</td>
<td>67.47</td>
<td>59.22</td>
<td>48.07</td>
<td>45.86</td>
</tr>
<tr>
<td>TMix</td>
<td>69.72</td>
<td>64.10</td>
<td>65.52</td>
<td>62.13</td>
<td>45.58</td>
<td>47.11</td>
</tr>
<tr>
<td>HYPMIX</td>
<td>69.25</td>
<td>63.10</td>
<td>64.36</td>
<td>60.48</td>
<td>48.10</td>
<td>41.28</td>
</tr>
<tr>
<td>ASMix</td>
<td>74.94</td>
<td>60.41</td>
<td>64.12</td>
<td>62.21</td>
<td>50.56</td>
<td>52.83</td>
</tr>
</tbody>
</table>

TABLE V: Performance (F1 (%)) of each category on THS in comparison with baselines.

<table>
<thead>
<tr>
<th>Model</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>98.58</td>
<td>79.94</td>
</tr>
<tr>
<td>EDA</td>
<td>96.33</td>
<td>87.37</td>
</tr>
<tr>
<td>AEDA</td>
<td>97.79</td>
<td>84.92</td>
</tr>
<tr>
<td>TMix</td>
<td>94.57</td>
<td>90.22</td>
</tr>
<tr>
<td>HYPMIX</td>
<td>96.37</td>
<td>88.61</td>
</tr>
<tr>
<td>ASMix</td>
<td>95.54</td>
<td>90.84</td>
</tr>
</tbody>
</table>

TABLE VI: Performance (F1 (%)) on three datasets after removing different parts of ASMix.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1(%)</th>
<th>R8</th>
<th>Cade12</th>
<th>THS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASMix</td>
<td>95.70</td>
<td>61.07</td>
<td>93.19</td>
<td></td>
</tr>
<tr>
<td>- smoothed</td>
<td>95.36</td>
<td>60.75</td>
<td>93.07</td>
<td></td>
</tr>
<tr>
<td>- threshold</td>
<td>94.51</td>
<td>59.80</td>
<td>92.86</td>
<td></td>
</tr>
<tr>
<td>- attention</td>
<td>94.87</td>
<td>59.78</td>
<td>91.96</td>
<td></td>
</tr>
<tr>
<td>- all</td>
<td>94.01</td>
<td>58.75</td>
<td>89.26</td>
<td></td>
</tr>
</tbody>
</table>

The IR of each class on these datasets is shown in Table III to V. The IR of the class keeps increasing because the model will be biased towards the majority class. BERT performs best in severe imbalanced categories in all three datasets. ASMix has a slight drop in categories with smaller IR on the F1 metric. It is natural for ASMix to trade off when ASMix focuses more on the minority class.

F. Varying the Number of Data

We evaluated our baselines and proposed methods using F1 with different fractions of the available training data ranging from 10 to 100 percent. The results on different text classification datasets are shown in Fig. 3. We show that ASMix consistently demonstrated the best performances when compared to different baseline models across three datasets with different fractions of the data. On Cade12, the best F1 without augmentation, 58.7%, was achieved using 100% of the training data. ASMix surpassed this number by achieving an F1 of 59% while only using 70% of the available training data.

G. Ablation Studies

We performed ablation studies to show the effectiveness of each component in ASMix. We measured the performance of ASMix by stripping each component each time and displayed the results in Table VI. We observed a drop in performance after removing each part, suggesting that all components in ASMix contribute to the final performance. Overall, the model performance dropped most significantly after removing the self-attention mechanism, which verified the effectiveness of the multi-token replacement strategy.

H. Parameter Studies

All parameter experiments use automatic mixed precision (AMP) of Pytorch, which saves memory and speeds up.

1) Hyper-parameter of the Beta Distribution $\alpha$: We varied the hyper-parameter $\alpha$ in $\{0.25, 0.5, 0.75, 1\}$. Fig. VII shows ASMix achieved optimal values on all three datasets when $\alpha$ was small. From the perspective of Beta distribution, smaller $\alpha$ leads to a lower threshold $\tau$ and $\lambda$ closer to 0.5, resulting...
in synthesized samples that are further away from the parent samples.

2) *Mixed Layer Set L*: Jawahar et al. [19] found that in BERT-based model, \{3, 4, 6, 7, 9, 12\} are the most informative layers. We chose to mix using different subsets of these layers to see which subset gave the best performance.

Our method achieves the best results on three datasets with \( L = \{6, 7, 9\} \), \( L = \{7, 9, 12\} \), and \( L = \{3, 4, 6, 7, 9, 12\} \), respectively. These layers mainly capture syntactic and semantic features that are very helpful for classification, such as the depth of the syntactic tree, the sequence of top level constituents in the syntax tree, sensitivity to word order, and the sensitivity to random replacement of a noun or verb. If we just mixup at the input and lower layers (\{0, 1, 2\}), there seemed no performance increase.

**V. CONCLUSION**

To alleviate the data imbalance problem, this work proposed an effective attention-based smooth data augmentation method, ASMix. This method augments the model input by using smoothed representations and varies minority class samples by adding discriminative information from other samples. Extensive experiments on three benchmark imbalanced text classification datasets prove the effectiveness of ASMix. For future directions, we plan to explore the effect of ASMix on semi-supervised tasks.

**REFERENCES**


An Optimal Spacing Approach for Sampling Small-sized Datasets for Software Effort Estimation

Samuel Abedu¹, Solomon Mensah²*, Frederick Boafo³, Eva Bushel² and Elizabeth Akuafum²
1Department of Computer Science and Software Engineering, Concordia University, Canada
2Department of Computer Science, University of Ghana, Accra - Ghana
3Department of Computer Science, Lancaster University Ghana

Abstract—Context: There has been a growing research focus in conventional machine learning techniques for software effort estimation (SEE). However, there is a limited number of studies that seek to assess the performance of deep learning approaches in SEE. This is because the sizes of SEE datasets are relatively small. Purpose: This study seeks to define a threshold for small-sized datasets in SEE, and investigates the performance of selected conventional machine learning and deep learning models on small-sized datasets. Method: Plausible SEE datasets with their number of project instances and features are extracted from existing literature and ranked. Eubank’s optimal spacing theory is used to discretize the ranking of the project instances into three classes (small, medium and large). Five conventional machine learning models and two deep learning models are trained on each dataset classified as small-sized using the leave-one-out cross-validation. The mean absolute error is used to assess the prediction performance of each model. Result: Findings from the study contradicts existing knowledge by demonstrating that deep learning models provide improved prediction performance as compared to the conventional machine learning models on small-sized datasets. Conclusion: Deep learning can be adopted for SEE with the application of regularisation techniques.

Keywords—Deep learning; Conventional Machine learning; Software effort estimation; Small-sized; Optimal spacing theory.

1. INTRODUCTION

A. Background and Motivation

In software engineering, the use of data for setting up predictive models is vital in estimating the effort required for projects. This data comprises features extracted from at least one software project and a collection of related sets for the various components of the project(s) to set up an SEE predictive model, such datasets are considered for the training and validation needs of the given model developed, before considering the estimated effort for new project instance(s).

It is known that deep learning works best for relatively large-sized datasets and has been considered effective for estimating the target or dependent variable when setting up classifiers and predictive models [1]. The key motivating question worth investigating is, can it work best for relatively small-sized dataset(s)? We seek to explore this issue in software engineering specifically in the domain of SEE. A leading question that also arises is that, how can we define small-sized data from a given dataset? We consider Eubank’s optimal spacing theorem [2] a plausible solution for addressing the small-sized issue raised and move on to perform a comparative study considering a sample of deep learning versus conventional machine learning techniques for our investigation.

B. Overview of Software Effort Estimation (SEE)

SEE deals with predicting the relevant effort in relation to project cost and resource allocation to enable timely production and delivery of software projects within budget [3]. When project effort is overestimated or underestimated, it can result in devastating consequences for the given company. For example, a company may face wastage of resources and contract loss in the case of effort overestimation, and poor project development quality or uncompleted projects in the case of underestimation. The estimation of software effort is done at the beginning of the project [4]. Accordingly, the precise effort of a software project is mostly determined when the project closes.

Deep learning methods have been successfully applied in various fields, including computer vision, speech recognition, software defect prediction, financial analysis but has not been adequately explored in SEE. Perhaps this is because, deep learning methods perform better on relatively large-sized datasets [5]. Yet, the ability to obtain relatively large datasets in SEE is a challenge because owners of such datasets are reluctant to share and provide mainly due to privacy issues. Thus, researchers use the available data considered small for estimating the effort [6]. There is the need to investigate a feasible approach for determining relatively small-sized data and investigating the performance of deep learning on small-sized data in SEE.

The remaining sections of the paper are organized as follows: Section 2 presents size of the SEE datasets considered for the study and the central limit theory. Section 3 presents the Eubank’s optimal spacing theorem. Section 4 details the methodological procedure, dataset description, data preprocessing, experimental setup, model selection and performance evaluation metrics. Section 5 presents the experimental results and discussion. Lastly, Section 6 concludes the study.

DOI reference number: 10.18293/SEKE2023-176
2. SIZE OF SEE DATASET & CENTRAL LIMIT THEOREM

Due to the relatively high cost of data collection, and unwillingness of companies to share software project data (due to privacy issues) [6], it has led to relatively small sizes of SEE datasets (L. Song et al., 2018). As a result of these challenges, historic or archival SEE data in the public domain are mostly used.

The definition of a small SEE dataset is ambiguous since different researchers have different perspectives of the definition (Song et al., 2019). Mostly, researchers failed to justify considering a particular type of dataset as small-sized. Evidence from the central limit theorem shows that a sample size greater than 30 is sufficient for the central limit theorem to apply. Hence, sample sizes less than or equal to 30 may be considered small. Yet, this statistical definition cannot be adopted for this study because SEE datasets with project instances greater than 30 have been described as small-sized (Song et al., 2019). This makes it imperative for this study to define the conditions for a small-size dataset that is appropriate for this investigation.

3. EUBANK’S OPTIMAL SPACING THEOREM

In a study by Eubank [2], a density quantile function method was introduced to address the optimal spacing selection problem for linear estimation over a given interval. This approach makes use of a quantile function over a given interval, \([p, q] \subseteq [0, 1]\) to generate a successful estimation of location and scale parameters for a censored set of order statistics. Optimal asymptotic spacings are generated from the quantile function. The optimal asymptotical spacing can be considered as a threshold value for the spacing selection over the interval. The density quantile function \(Q(u)\) is defined in (1).

\[
Q(u) = F^{-1}(u), \ 0 \leq u \leq 1
\]  

where \(F\) is considered as the censored set distribution function and \(u\) is regarded as the location parameter for defining the asymptotic optimal spacing.

The theorem was considered to discretize the rankings of the studied datasets into three main classes. The theorem uses the density quantile function to discretize the rankings of the frequency instances per each dataset into three classes. Class 1 is regarded as the first-class (\(Q_1\)), class 2 as the second-class (\(Q_2\) and \(Q_3\)), and class 3 as the third-class (\(Q_4\)). Datasets ranked with total number of instances less than or equal to first class are classified as small-sized datasets. Alternatively, all datasets with ranks greater than or equal to the third class are classified as large-sized datasets, and the remainder as medium-sized datasets, that is in-between the first and third classes.

3.1. EMPIRICAL FRAMEWORK

A. Dataset Description

The study selected 22 publicly available datasets from the PROMISE and GitHub repositories. The ranking of the data sizes was based on the total number of instances for the studied datasets. The project dataset with the minimum number of instances was ranked the lowest (one) and the project with the maximum number of instances was ranked highest (22).

Six out of the 22 datasets were classified as small-sized based on the classification scheme provided in Section 3. Thus, these 6 datasets were adopted for the study’s empirical and comparative analysis. The number of project instances, project features, description of project type, the mean and standard deviation of all 6 datasets (Albrecht, Atkinson, Cosmic, Kemerer, Finnish, and Telecom1) are presented in Table 1. A maximum threshold of 43 project instances based on Eubank’s optimal spacing theory is defined for classifying a given SEE dataset as small-sized.

The discretization scheme classified the Java software project dataset [6], Tukutuku dataset, Bietak dataset, dataset I [12], USP05 dataset and China dataset as large-sized. Similarly, a minimum threshold of 147 project instances is defined for classifying a dataset as large-sized. Datasets with project instances within the range of 44 to 146 inclusive are classified as medium-sized. Thus, 10 datasets, namely, Miyazaki94, Miyazaki, Qi et al.’s [6] Web data, Nasa, Cocomod1, Maxwell, Lopez-Martin’s [12] Data 2, Desharinai, Nasaz93 and Kitchenham dataset.

In addition to the small-sized datasets, two relatively large-sized datasets, namely the ISBSG and China datasets [13] were also used. These relatively large-sized datasets were added to test the assertion that, deep learning models perform better on larger datasets than conventional machine learning models for SEE. The ISBSG dataset release 10, which contains 4106 cross-organisational projects was used.

Table 1. Description of selected datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Instances</th>
<th>Features</th>
<th>Description</th>
<th>Mean (Effort)</th>
<th>Std (Effort)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albrecht</td>
<td>24</td>
<td>7</td>
<td>IBM DP Services Project</td>
<td>21.9</td>
<td>28.4</td>
</tr>
<tr>
<td>Atkinson</td>
<td>16</td>
<td>12</td>
<td>Builds to a large telecommunications product at U.K. company X</td>
<td>456.1</td>
<td>241.1</td>
</tr>
<tr>
<td>Cosmic</td>
<td>42</td>
<td>10</td>
<td>N/A</td>
<td>4965.5</td>
<td>8457.1</td>
</tr>
<tr>
<td>Finnish</td>
<td>38</td>
<td>6</td>
<td>Data collected by the TIEKE organization from IS projects from nine different</td>
<td>7678.3</td>
<td>7135.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Finnish companies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kemerer</td>
<td>15</td>
<td>5</td>
<td>Large business application</td>
<td>219.2</td>
<td>263.1</td>
</tr>
<tr>
<td>Telecom1</td>
<td>18</td>
<td>2</td>
<td>Enhancement to a U.K. telecommunication product</td>
<td>284.3</td>
<td>264.7</td>
</tr>
<tr>
<td>ISBSG</td>
<td>4106</td>
<td>105</td>
<td>Cross-organisational projects compiled by the ISBSG</td>
<td>5925.5</td>
<td>20685.9</td>
</tr>
<tr>
<td>China</td>
<td>499</td>
<td>17</td>
<td>Data collected from various software firms</td>
<td>3921.0</td>
<td>6480.9</td>
</tr>
</tbody>
</table>
Cross-organizational projects are heterogeneous in nature and do not share common characteristics like development policy, development team and programming skills. As a result, the 642 projects from the Communication organization type were selected for this study. Similar to the study by Mensah et al. [14], the study selected the project size, development type, language type and development platform as the features and the normalised development effort as the dependent variable.

B. Experimental Setup

Seven prediction models were set up with each model trained on the selected datasets. The models were trained based the leave-one-out cross-validation (LOOCV) approach, and the Mean Absolute Error (MAE) was used to evaluate the performance. The studied datasets were pre-processed using the data pre-processing approach described in Section 4C. The log transformation technique was applied to the datasets to minimise the effect of skewness. Feature selection was also performed to select relevant features from the data for the model training. The early stopping technique was applied to reduce the overfitting of the deep learning models. This study implemented the early stopping regularisation technique as used in the work of Kalichanin-Balich and Lopez-Martin [15] to reduce overfitting and improve the performance of the deep learning models. In addition, dropout was added after each hidden layer to also reduce overfitting.

C. Data Pre-processing

There is the need for high quality data to improve the quality of mining results. Thus, pre-processing the data to achieve high quality is very important when training and validating machine learning models. To achieve high data quality (i.e., resolve issues of missing data values, outliers, and influential data) the datasets were pre-processed before setting up the prediction models as follows: An in-depth analysis of the datasets was performed to address missing data values. Columns of each project instance were examined for null or NaN values. The ISBSG was the only dataset that contained missing values. Project instances in the ISBSG that had missing entries for the selected features were eliminated. Data point with Cook’s distance measures three times more than the mean of Cook’s distance values were considered to be outliers. Whereas data points whose DFFITS were greater than one was considered influential [14].

D. Model Selection

This section discusses the models selected for this study and the hyperparameter tuning approach used. The models are the Automatically Transformed Linear Model (ATLM), Bayesian Network (BN), ElasticNet (ENR) regression, Support Vector Machine (SVM), Artificial Neural Network (ANN), Deep Neural Network (DNN) and Long-Short Term Memory (LSTM). The hyperparameters of the models were fine-tuned using the Bayesian optimization approach. This approach models the generalisation performance of a prediction model as a Gaussian process and maintains a posterior distribution as observations are made on the results from running the model with different hyperparameters. The Bayesian optimization algorithm picks the hyperparameter values for the next experiment based on the Gaussian process upper confidence bound (UCB).

1) Deep Learning Models

A Deep learning model simulates human brain processes. DNN is made of an input layer, an output layer and multiple hidden layers which are made of hidden nodes called neurons. Note that for ANN, there is a maximum of one hidden layer. Each neuron receives input and process it solving a non-linear function of the inputs by assigning weights. Taking the weights assigned to the neuron as $w_{ij}$ at the start of training. The weights are updated after each training epoch $(t)$ to $w_{ij}^{(t+1)}$ where the change in weights $(\Delta w_{ij}^{(t+1)})$ is defined in (2).

$$\Delta w_{ij}^{(t+1)} = \gamma \Delta w_{ij} - \alpha \frac{\partial L(.)}{\partial w_{ij}}$$  \hspace{1cm} (2)

where $\gamma$ is the learning momentum, $\alpha$ is the learning rate and $L(.)$ is the loss function. This study adopted the RMSProp optimization algorithm for model training and the optimal learning rate $(\alpha)$ was identified using the Bayesian Optimization algorithm. The Parametric Rectified Linear Unit (PRelu) which improves the accuracy of models over the Relu was used. PRelu preserves the properties of linear models, this makes it easier to be optimized. The PRelu is defined in (3).

$$PRelu(x) = \max(ax,x)$$  \hspace{1cm} (3)

where $a$ is the slope parameter learned by the model through backpropagation. Dropout and early stopping were applied to reduce overfitting. The efficient dropout was identified by using Bayesian Optimization. This study adopted the feed-forward neural network with the backpropagation of errors. The number of hidden layers used in this study varied between two and five. Also, Bayesian optimization was used to identify the optimum learning rate $\alpha$ and the dropout for the models. The LSTM has been demonstrated to be efficient in modelling time-series data. Considering that software project starts and ends at distinct time points, software effort data can be classified as pseudo time series data and hence makes the LSTM suitable for this study. Also, considering that no study had considered it in traditional effort estimation, it was adopted. The LSTM is a variant of recurrent neural networks that solves the vanishing gradient problem [16]. It uses a loop in the network that allows information to persist in the network. Given a data $(X,Y)$ with $(x_1,...,x_n) \in X$ and outputs $(y_1,...,y_n) \in Y$. At step $t$ the LSTM reads input $x_t$, the hidden state $h_{t-1}$ and the previous memory $c_{t-1}$ to compute hidden state $h_t$. The hidden state $h_t$ is used to predict the output $y_t$. The memory cell is the element that stores the accumulated information over time. The LSTM model was also set up with two LSTM layers and the glorot uniform kernel initializer was used. The PRelu activation function was used for all layers in the network.
Dropout and early stopping were applied to reduce overfitting. The dropout and the learning rate ($\alpha$) of the model were fine-tuned using the Bayesian optimization. The optimum values for the dropout and the learning rate ($\alpha$) for the ANN, DNN and LSTM models are presented in Table 2.

2) Conventional Machine Learning Models

The ATLM is a baseline model for comparative studies in SEE. It is based on a multiple linear regression, and it performs automatic data pre-processing such as log and square-root transformations on the data and also compares the skewness in the dependent and independent variables. It uses the least skewed data for model construction and effort prediction.

The Bayesian network (BN) is a graphical model that encodes the probabilistic relationship between related variables. BN is determined by a pair as defined in (4).

$$BN = (G, P)$$

(4)

where $G$ is a directed acyclic graph with nodes $X_i$ and $P$ is the local probability of all variables in the network. For instance, given node $X_1$ connected to node $X_2$ and node $X_3$ connected to node $Y$, then the conditional probability of finding $Y$ in (5).

$$P(Y/X_2, X_3) = P(Y/X_2)$$

(5)

The joint probability of each variable satisfies Markov’s condition that each variable $X_i$ is conditionally independent of the set of its non-descendants. The distribution is factorised in (6).

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|\pi(X_i))$$

(6)

where $\pi(X_i)$ is the set of nodes directly connected to $X_i$. The Bayesian network takes into account the probability effect of each input variable on the dependent variable.

The ElasticNet (ENR) performs regularisation and feature selection to get rid off highly correlated estimator variables before developing the model. Given a set of input data with $N$ observation pairs $(x_i, y_i)$ and an approximated regression function $E(Y/X = x) = \beta_0 + x^T \beta$, the ENR finds the optimal $\beta$ by solving the problem defined in (7).

$$\hat{\beta} = \min_{(\beta_0, \beta)} \left[ \frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \beta)^2 + \lambda P_\alpha(\beta) \right]$$

(7)

where $P_\alpha(\beta)$ is the elastic net penalty given by

$$P_\alpha(\beta) = \left[ \frac{1}{2} (1 - \alpha) \beta_0^2 + \alpha |\beta| \right]$$

For $i = 1, ..., p$. The $P_\alpha$ is used in finding highly correlated input variables. $\alpha$ is the hyperparameter tuned for the study. The value of $\alpha$ resulting in the best performance for each dataset is presented in Table 2.

The SVM is a supervised learning approach for classification that has been extended for regression problems as $\epsilon$-SVM. The $\epsilon$-SVM uses Lagrange multipliers to find a function $f(X_i)$ that has a deviation of at most $\epsilon$ from the dependent variable and then produces a final solution from a combination of cases from the dataset. The kernel function of the SVM allows the $\epsilon$-SVM to handle datasets that are of non-linear and complex. The kernel is treated as a hyperparameter. The best performing kernel for the SVM on each dataset is recorded in Table 2.

### Table 2. Optimal Hyperparameter values for each Model

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dropout</th>
<th>Learning Rate</th>
<th>GAM</th>
<th>Dropout</th>
<th>Learning Rate</th>
<th>GAM</th>
<th>Dropout</th>
<th>Learning Rate</th>
<th>GAM</th>
<th>Dropout</th>
<th>Learning Rate</th>
<th>GAM</th>
<th>Dropout</th>
<th>Learning Rate</th>
<th>GAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albrechet</td>
<td>0.0294</td>
<td>0.0327</td>
<td>0.0732</td>
<td>0.0092</td>
<td>0.092</td>
<td>0.035</td>
<td>0.0996</td>
<td>RBF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atkinson</td>
<td>0.2438</td>
<td>0.0064</td>
<td>0.0732</td>
<td>0.0092</td>
<td>0.169</td>
<td>0.028</td>
<td>0.1956</td>
<td>Sigmod</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cosmic</td>
<td>0.2081</td>
<td>0.072</td>
<td>0.0732</td>
<td>0.0992</td>
<td>0.209</td>
<td>0.088</td>
<td>0.0285</td>
<td>Polynomial</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finnish</td>
<td>0.2081</td>
<td>0.072</td>
<td>0.0720</td>
<td>0.102</td>
<td>0.087</td>
<td>1.3e-07</td>
<td>Linear</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kemerer</td>
<td>0.2081</td>
<td>0.072</td>
<td>0.0732</td>
<td>0.0992</td>
<td>0.216</td>
<td>0.057</td>
<td>0.0074</td>
<td>Linear</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Telecom</td>
<td>0.2081</td>
<td>0.072</td>
<td>0.0732</td>
<td>0.0992</td>
<td>0.208</td>
<td>0.055</td>
<td>0.0157</td>
<td>Sigmod</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISRSG</td>
<td>0.2000</td>
<td>0.001</td>
<td>0.2000</td>
<td>0.001</td>
<td>0.200</td>
<td>0.001</td>
<td>0.0001</td>
<td>Polynomial</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>China</td>
<td>0.2000</td>
<td>0.001</td>
<td>0.2000</td>
<td>0.001</td>
<td>0.200</td>
<td>0.001</td>
<td>0.0001</td>
<td>Linear</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### E. Performance Measure

MAE is a robust, effective and reliable performance metric for assessing SEE models and measuring the average error magnitude. The MAE is an unbiased error estimator. It has been considered in the evaluation of estimation accuracy and thus it was considered for this study. The MAE is defined in (8).

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |E_{A_i} - E_{E_i}|$$

(8)

where $E_{A_i}$ is the actual effort of a project case or instance and $E_{E_i}$ is the estimated effort of a project instance with $n$ number of instances. A single performance evaluation measure was used to avoid the ambiguity associated with the conclusion instability issue.

Cliff’s $d$ effect size was used to assess the practical significance of the model. The effect size is less affected by the sample size than other statistical significance tests such as Yuen’s [18]. The effect size provides an objective measure of the importance of an experimental effect and in this empirical study, it was used to measure the magnitude of the difference between the actual effort ($y$) and the predicted effort ($\hat{y}$). It is defined in (9).

$$\delta = \frac{\text{COUNT}(y_1 > \hat{y}_1) - \text{COUNT}(y_1 < \hat{y}_1)}{nm}$$

(9)

where $y_1$ is the actual effort and the $\hat{y}_1$ is the predicted effort of each dataset. The number of project cases in the actual and predicted classes are denoted by $n$ and $m$ respectively.

Kampenes et al. [18] provides interpretation for the cliff’s delta effect size as follows: (1) negligible refers to $\delta < 0.112$, small is $0.112 \leq \delta < 0.276$, medium is $0.276 \leq \delta < 0.427$ and large refers to $\delta \geq 0.427$. This study makes use of an effect size threshold of negligible ($\delta < 0.112$), with the argument, that the difference in magnitude between the actual and predicted effort should be negligible.
5. RESULT AND DISCUSSION

The result as presented in Table 3 show that the deep learning models (DNN and LSTM) recorded the best prediction accuracy for five datasets. The ANN, which is a shallow neural network recorded the best prediction accuracy for two datasets and the baseline ATLM also recorded the best prediction accuracy for one dataset. The ANN had the best prediction accuracy for the Albrecht dataset with negligible practical significance difference between the predicted and actual efforts. DNN and LSTM models also produced prediction accuracies better than the conventional machine learning models on the Albrecht dataset. The results of the models using the Atkinson dataset as presented in Table 3 shows the DNN model as having the best prediction accuracy. The DNN recorded a mean absolute error (MAE) of 0.0018 with a 0.0102 Cliff’s δ effect size (signifying low practical significance difference). The LSTM and ANN also produced an impressive prediction accuracy with MAE of 0.002 and 0.0036 respectively. The results for the Atkinson dataset showed that the deep learning models outperform the conventional machine learning models. This accounted for the conventional machine learning models recording higher MAEs (poor performance).

For the Cosmic dataset, the LSTM model recorded the best prediction accuracy with an MAE of 0.0075. Again, the ANN and DNN also achieved high prediction accuracy as compared to the conventional machine learning models. The high MAE measured for the conventional machine learning was as a result of more variations in the absolute values of the model’s prediction on the cosmic dataset.

For the Finnish dataset, the conventional machine learning models outperformed the deep learning models. The baseline ATLM recorded an MAE of 0.0006. The Bayesian Network (BN), ElasticNet and the SVM recorded MAE of 0.0014, 0.0021 and 0.001 respectively. The DNN, LSTM and ANN also recorded MAE of 0.0117, 0.0092 and 0.0125 respectively. The DNN proved to be the best model for the Kemeler dataset with an MAE of 0.0088 and Cliff’s δ effect size of 0.0306. The LSTM and ANN model recorded MAE of 0.0239 and 0.0143 respectively. The SVM was the best conventional machine learning model with an MAE of 0.1614. The performance of the models on the Telecom dataset showed the LSTM as the best. The LSTM model recorded an MAE of 0.0029 with a 0.0133 Cliff’s δ effect size. The DNN and ANN models also achieved MAE of 0.0086 and 0.0034 respectively. The SVM with an MAE of 0.1279 was the best performing conventional machine learning model on the Telecom dataset. The baseline ATLM produced an MAE of 0.206. For the ISBSG, which is a large-sized data, the artificial neural network recorded the best performance with an MAE of 0.0038 and Cliff’s delta effect size of 0.0001. The two deep learning models recorded better performance than the conventional machine learning models.

The DNN and LSTM models recorded MAEs of 0.0049 and 0.0044 respectively compared to the MAEs of the ATLM, Bayesian network, Elastic Net and SVM which were 0.0136, 0.0136, 0.0136 and 0.3530 respectively.

Also, for the other large-sized dataset, the China dataset, the LSTM recorded the best MAE, 0.0048 with Cliff’s delta effect size of 0.0017. The DNN model also recorded an MAE of 0.0049 with Cliff’s delta effect size of 0.0001. These two deep learning models with the artificial neural network outperformed the conventional machine learning models on this dataset. The conventional machine learning models recorded MAEs of 0.0061, 0.0068, 0.0068 and 0.1738 as shown in Table 3. The magnitude of Cliff’s δ effect size measurement for all models were negligible (δ < 0.112) as defined by Kampenes et al. [18]. This means there was little difference between the magnitude of the actual effort values and the predicted effort estimations of each model.

6. CONCLUSION AND FUTURE WORK

In this study, a threshold for defining a small-sized dataset using Eubank’s optimal spacing theory is introduced in SEE. Twenty-two datasets were selected from recent SEE studies. The classification scheme based on these 22 datasets classified six datasets as small-sized and defined a threshold of 43 project instances for a small-sized dataset. The six datasets were selected for the empirical study in addition to two large-sized datasets. The classification scheme was constructed using Eubank’s optimal spacing theorem [2]. This is due to the advantage of splitting a given interval into optimal sizes or thresholds.

The empirical study investigated the prospects of deep learning in SEE by comparing the performance of the ATLM, SVM, Bayesian network, ElasticNet, ANN, DNN and LSTM models. The leave-one-out cross-validation (LOOCV) was applied in

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>ATLM</th>
<th>DNN</th>
<th>LSTM</th>
<th>ANN</th>
<th>BN</th>
<th>ENR</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albrecht</td>
<td>MAE</td>
<td>0.352</td>
<td>0.014</td>
<td>0.024</td>
<td>0.006</td>
<td>0.239</td>
<td>0.248</td>
<td>0.186</td>
</tr>
<tr>
<td></td>
<td>Cliff’s δ</td>
<td>0.004</td>
<td>0.017</td>
<td>0.010</td>
<td>0.031</td>
<td>0.010</td>
<td>0.010</td>
<td>0.084</td>
</tr>
<tr>
<td>Atkinson</td>
<td>MAE</td>
<td>0.052</td>
<td>0.002</td>
<td>0.002</td>
<td>0.004</td>
<td>0.038</td>
<td>0.034</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>Cliff’s δ</td>
<td>0.000</td>
<td>0.010</td>
<td>0.000</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td>Cosmic</td>
<td>MAE</td>
<td>0.549</td>
<td>0.020</td>
<td>0.008</td>
<td>0.013</td>
<td>0.516</td>
<td>0.475</td>
<td>0.467</td>
</tr>
<tr>
<td></td>
<td>Cliff’s δ</td>
<td>0.001</td>
<td>0.005</td>
<td>0.001</td>
<td>0.002</td>
<td>0.002</td>
<td>0.001</td>
<td>0.007</td>
</tr>
<tr>
<td>Finnish</td>
<td>MAE</td>
<td>0.004</td>
<td>0.012</td>
<td>0.009</td>
<td>0.013</td>
<td>0.001</td>
<td>0.002</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>Cliff’s δ</td>
<td>0.002</td>
<td>0.002</td>
<td>0.007</td>
<td>0.002</td>
<td>0.002</td>
<td>0.009</td>
<td>0.002</td>
</tr>
<tr>
<td>Kemeler</td>
<td>MAE</td>
<td>0.206</td>
<td>0.009</td>
<td>0.024</td>
<td>0.014</td>
<td>0.196</td>
<td>0.173</td>
<td>0.161</td>
</tr>
<tr>
<td></td>
<td>Cliff’s δ</td>
<td>0.000</td>
<td>0.031</td>
<td>0.031</td>
<td>0.020</td>
<td>0.020</td>
<td>0.020</td>
<td>0.000</td>
</tr>
<tr>
<td>Telecom</td>
<td>MAE</td>
<td>0.206</td>
<td>0.009</td>
<td>0.003</td>
<td>0.003</td>
<td>0.216</td>
<td>0.199</td>
<td>0.128</td>
</tr>
<tr>
<td></td>
<td>Cliff’s δ</td>
<td>0.004</td>
<td>0.004</td>
<td>0.013</td>
<td>0.023</td>
<td>0.004</td>
<td>0.013</td>
<td>0.031</td>
</tr>
<tr>
<td>ISBSG</td>
<td>MAE</td>
<td>0.014</td>
<td>0.005</td>
<td>0.004</td>
<td>0.004</td>
<td>0.014</td>
<td>0.014</td>
<td>0.350</td>
</tr>
<tr>
<td></td>
<td>Cliff’s δ</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
<td>0.001</td>
<td>0.006</td>
</tr>
<tr>
<td>China</td>
<td>MAE</td>
<td>0.006</td>
<td>0.005</td>
<td>0.005</td>
<td>0.007</td>
<td>0.007</td>
<td>0.007</td>
<td>0.174</td>
</tr>
<tr>
<td></td>
<td>Cliff’s δ</td>
<td>0.000</td>
<td>0.000</td>
<td>0.002</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

The values in bold represent the best performance measure for each dataset across the learners – ATLM, DNN, LSTM, ANN, BN, ElasticNet (ENR) and SVM

δ - Cliff’s delta effect size
this study for the training and validation needs of the selected models, and performance evaluation was done using MAE. The results showed the selected deep learning models outperformed the conventional machine learning models on the Atkinson, Cosmic, Kemerer and Telecom datasets. The ANN, which is a shallow neural network, outperformed the deep learning models on the Albrecht dataset, and likewise the ATLM on the Finnish dataset. Also, findings from the study show that deep learning models have better performance on large-sized SEE datasets than conventional machine learning models.

The study concluded from the results that deep learning should be adopted for software effort estimation. However, it recommends techniques like dropout and early stopping to reduce overfitting in the deep learning models.

In future, the empirical study will be extended to evaluate the computational cost of running the deep learning and conventional machine learning models in the software engineering field. This will guide researchers and practitioners on the trade-offs that can be made in choosing either a conventional machine learning or deep learning model.

REFERENCES


Heterogeneous Multi-Agent Communication Learning via Graph Information Maximization

Wei Du  
School of Computer Science and Technology  
China University of Mining and Technology  
Xuzhou, China  
1394471165@qq.com

Shifei Ding  
School of Computer Science and Technology  
China University of Mining and Technology  
Xuzhou, China  
dingsf@cumt.edu.cn

Abstract—Communication learning is an effective way to solve complicated cooperative tasks in multi-agent reinforcement learning (MARL) domain. Graph neural network (GNN) has been widely adopt for learning the multi-agent communication and various GNN-based MARL methods have emerged. However, most of these methods are not specially designed for heterogeneous multi-agent scenarios, where agents have heterogeneous attributes or features based on different observation spaces or action sets. Without effective processing and transmission of heterogeneous feature information, communication learning will be useless and even reduce the performance of cooperation. To solve this problem, we propose a communication learning mechanism based on heterogeneous GNN and graph information maximization to learn effective communication for heterogeneous agents. Specifically, we use heterogeneous GNN for learning the efficient message representations, which aggregate the local feature information of neighboring agents. Furthermore, we maximize the mutual information (MI) between message representations and local values to make efficient use of information. Besides, we present a MARL framework that can flexibly integrate the proposed communication mechanism with existing value factorization methods. Experiments on various heterogeneous multi-agent scenarios demonstrate the effectiveness and superiority of the proposed method compared with baselines.

Index Terms—Communication learning, Multi-agent reinforcement learning, Mutual information, Graph neural network.

I. INTRODUCTION

In recent years, multi-agent reinforcement learning (MARL) has seen tremendous growth and attracted wide attention [1]. The paradigm of centralized training and decentralized execution (CTDE) is popular and widely used in MARL because it can address scalability issues and partial observability limitations of MARL [2]. Most of the CTDE-based MARL methods can be roughly divided into value factorization methods [3], [4], [5] and communication learning methods [6], [7], which provide different solutions for further exploiting CTDE paradigm. The value factorization methods factorize the global value function into the set of the local individual value function of each agent to further tackle the scalability issue. The communication learning methods enable agents to share important information in the decentralized execution period, which can further promote action coordination.

Recently, graph neural network (GNN), as an efficient representation learning method [8], has been widely utilized to build communication learning mechanism of MARL, which generally regards agents as nodes in the graph, with the communication channels corresponding to edges. Many state-of-the-art MARL methods fall into this GNN-based communication paradigm [9], [10]. However, most GNN-based communication learning methods are not specially designed for heterogeneous scenarios, where agents have different observation spaces or action sets. Therefore, these methods can not effectively process and transmit heterogeneous feature information, which leads to inefficient communication learning and affects action coordination.

To solve these problems, we present a Communication Learning mechanism of multi-Agent Reinforcement learning (CLAR) for heterogeneous scenarios. The proposed mechanism utilizes heterogeneous GNNs to model the heterogeneous agents and fuse feature information of neighboring agents to obtain high-level message representations. Besides, the proposed mechanism leverages mutual information (MI) optimization to obtain high-quality message representations for action coordination. Furthermore, we present a MARL framework that integrates the value factorization and the proposed communication learning mechanism. This framework can maintain the advantages of the stability and scalability of the value factorization methods, and promote better action coordination between agents by effectively processing and utilizing heterogeneous feature information. The following are the primary contributions of the proposed method,

1) We present a MARL framework that integrates communication learning mechanism and value factorization methods for heterogeneous scenarios, which solved the communication learning challenge of heterogeneous scenarios and the action discoordination issue of value factorization methods.

2) We first introduce the MI between the local values and the message representations in MARL. We use the MI maximization to learn the most valuable and expressive information from different classes of agents for better action coordination.

3) We design the heterogeneous GNN to learn heterogeneous multi-agent communication, which efficiently models the heterogeneous scenarios and achieves the fusion and transmission of heterogeneous information.
II. RELATED WORK

A. GNN-based MARL

Recently, learning multi-agent communication via GNN in MARL has attracted popular attention. Jiang et al. [9] first extends GNN to MARL for multi-agent communication learning. Das et al. [10] leverages GNN with soft attention mechanism to learn whom to receive messages and what messages to pass. Sheng et al. [11] utilizes the hierarchical GNN to achieve effective communication learning by sharing information among agents and groups. Ryu et al. [12] presents a hierarchical attention mechanism based on GNNs, which models the relationships between agents effectively. Liu et al. [13] utilizes a two-stage attention mechanism to model the complete graph for communication learning. Niu et al. [14] proposes an attentional GNN to tackle the challenges of how to process information and when to communicate.

The existing GNN-based MARL methods have achieved efficient communication learning by modeling interactions or relationships between agents. However, most of these methods are not specially designed for heterogeneous scenarios, where agents have heterogeneous attributes or features based on different observation spaces or action sets. Although some works attempt to use heterogeneous GNNs to learn communication in heterogeneous scenarios [15], [16], these works do not further optimize the high-level message representation, resulting in communication learning less effective. Different from them, the proposed method utilizes MI optimization to obtain high-quality message representations for action selection.

B. Graph Convolutional Network

Graph Convolutional Network (GCN) as a popular GNN module generally utilizes the information passing between the graph nodes to learn the structural dependency between nodes [8]. Concretely, each node aggregates the feature of adjacent graph nodes to compute a new high-level feature vector, the feature aggregation procedure is shown in Eq.(1).

\[ h_i' = \sigma \left( \sum_{b \in N(i)} \omega h_b / d_{ib} \right) \]  

where \( h_i' \) represents the aggregated feature vector of node \( i \) , \( \sigma(\cdot) \) denotes the activation function and \( \omega \) denotes the learnable weights. \( b \in N(i) \) contains the immediate neighbor nodes of node \( i \), where \( b \) represents the index of the neighbors. \( d_{ib} \) denotes the normalization term, which has several options and the common one is \( \sqrt{|N(i)|N(b)} \). After feature aggregation through several layers, the high-level feature representation of node \( i \) can integrate the structural information of nodes reachable from graph node \( i \).

In our work, we use an efficient enhancement of Eq. (1), which replace \( d_{ib} \) with the attention coefficients \( \alpha_{ib} \) as follows:

\[ \alpha_{ib} = \text{softmax}_b (\sigma' (\bar{a}^T \omega h_i || \omega h_b)) \]  

where \( \sigma' \) utilizes the LeakyReLU nonlinearity function, \( \bar{a} \) represents the learnable weight, || denotes concatenation operation. The softmax function is utilized to normalize the coefficients over all neighbor nodes \( b \).

III. METHODOLOGY

In this section, we introduce the proposed MARL framework and the proposed communication learning mechanism based on the heterogeneous GCN and MI optimization.

A. Problem formulation

In our work, we formulate the heterogeneous multi-agent issue as the Heterogeneous Multi-Agent POMDP represented by \( G = \langle C, I, \{ S^c \}_{c \in C}, \{ A^c \}_{c \in C}, \{ O^c \}_{c \in C}, R \rangle \) [16]. \( C \) represents the set of all classes of agents in the heterogeneous scenarios and the index \( c \in C \) represents the class that the agent belongs to, the total number of classes is denoted as \( n \). \( I = \sum_{c \in C} f^c \) represents the total number of collaborating agents in which \( f^c \) denotes the number of agents that belong to class \( c \). \( \{ S^c \}_{c \in C} \) denotes the state space in which \( S^c \) represents the joint state of agents of class \( c \). \( S^c = [s^c_t]_{t=1}^{T^c} \) and \( s^c_t \) represents the state of agent \( i \) of class \( c \). \( \{ A^c \}_{c \in C} \) represents the action space in which \( A^c \) represents the joint action space of agent \( i \) of class \( c \). \( A^c = [a^c_t]_{t=1}^{T^c} \) and \( a^c_t \) represents the action of agent \( i \) that belongs to class \( c \). \( \{ O^c \}_{c \in C} \) represents the observation space where \( O^c \) represents the observation space of agents of class \( c \). Each agent \( i \) of class \( c \) obtains a partial observation \( o^c_i \in O^c \) and takes an action \( a^c_i \in A^c \), which forms the joint action \( (a^c_i, o^c_i) \in C \). Then agent \( i \) can obtain an immediate shared reward \( R(s, a) \), which encourages cooperated behavior among agents. The target of all the agents is to learn the optimal joint action-value function \( Q_{\text{tot}}(\tau, a) = \mathbb{E}_{s, a} \sum_{t=0}^{\infty} \gamma^t R(s, a) \), where \( \tau \) represents the joint action-observation history, and \( \gamma \) represents the temporal discount factor.

B. Overall Framework

The overall framework of the proposed method is shown in Fig. 1, which contains 3 modules: feature encoding module, communication learning module, and value decomposition module. For agent \( i \), it receives the local observation \( o^c_i \) and then utilizes Multi-Layer Perceptron (MLP) and Gated Recurrent Unit (GRU) to process the local observation and produce the feature \( h_i \). Then \( h_i \) is fed to encoder to generate type-specific features \( \{ h^c_i, h^c_{i-1}, \ldots, h^c_0 \} \) for communication. The communication module is built by the heterogeneous GCN to pass the heterogeneous feature information among agents and learn specific communication policies based on agent types. By stacking multiple heterogeneous GCN layers, the high-level embedding \( m_i \) of agent \( i \) can be extracted through multiple rounds of communication.

In value decomposition module, the local individual action-value function \( Q_i(\tau, a_i, m_i) \) is calculated based on local observation history \( \tau_i \) and feature messages \( m_i \) received from the communication learning module. Then the local Q values obtained by all the agents are input into a mixing network to generate an estimation of the global value. Besides, we utilize mutual information optimization to further strengthen the correlation between the communication learning module and value decomposition module. The proposed communication mechanism can be fused with any factorization
methods under the paradigm of centralized training and decentralized execution. During the decentralized execution phase, the agents can communicate and take actions in a decentralized manner based on the communication learning module.

C. Communication Learning

In this section, we utilize the heterogeneous GCN to build the communication learning mechanism. The overall framework of the communication module is shown in Fig. 2, which contains the message sender procedure and receive procedure. For simplicity and universality, we consider a heterogeneous scenario with three types of agents, $C = \{F, K, L\}$ and take the agent of class $F$ as an example. For agent $f$ of class $F$, its obtained feature $h_f$ is processed by different weight matrix and sent to other agents during the sender procedure. On the one side, $h_f$ is processed utilizing a class-specific weight matrix $w_F \in \mathbb{R}^{d \times d'}$, where $d$ and $d'$ represents the dimension of the input feature and the output feature, respectively. On the other side, $h_f$ is processed by heterogeneous edgetypes utilizing the edgetype-specific weight matrix $w_{\text{Edgetype}} \in \mathbb{R}^{d'' \times d'}$, where the dimension of the output feature of the agent that agent $f$ sent messages to.

For example, $F \rightarrow K$ represents the edgetype from the agent of class $F$ to the agent of class $K$. $h_k^F = w_{F \rightarrow K} h_f$ denotes the feature processed by edgetype-specific weight matrix $w_{F \rightarrow K}$ that from agent $f$ of class $F$ to the any agent of class $K$. Then the obtained feature $h_k^F$ are sent to any agent of class $K$. During the message receive phase, for each edge type that the agent $f$ of class $F$ is connected to other agents, we utilize the heterogeneous GCN with attention mechanism to obtain the per-edge-type aggregation feature. It is obtained by weighted calculation of messages received by neighbor agents along the same edge type with $\alpha_{\text{Edgetype}}$, which denotes normalized attention coefficients.

Then, the aggregation embeddings are integrated with transformed embedding $w_F h_f$ to compute the output message embedding. Therefore, for agent $f$ of class $F$, the message aggregation equation can be represented as follows, where $N_f(F)$, $N_f(K)$ and $N_f(L)$ represents the neighbor agents that belongs to class $F$, $K$ and $L$, respectively.

$$m_f = \sigma \left( w_F h_f + m_k^F + m_l^F + m_k^F \right)$$

where $m_k^F = \sum_{f' \in N_f(F)} \alpha_{ff'} h_{f'}^F$, $m_k^F = \sum_{k \in \mathbb{N}_f(K)} \alpha_{fk} h_k^F$, and $m_l^F = \sum_{l \in \mathbb{N}_f(L)} \alpha_{fl} h_l^F$. To consider heterogeneous communication, we utilize Eq.(3)-Eq.(5) to compute the attention coefficient $\alpha_{ff'}$, $\alpha_{fk}$, $\alpha_{fl}$ in the message representations.

$$\alpha_{ff'} = \text{softmax}_{f'} \left( \sigma' \left( \bar{a}^T \left[ w_F h_f || w_{F \rightarrow F} h_{f'} \right] \right) \right)$$

$$\alpha_{fk} = \text{softmax}_k \left( \sigma' \left( \bar{a}^T \left[ w_F h_f || w_{K \rightarrow F} h_k \right] \right) \right)$$

$$\alpha_{fl} = \text{softmax}_l \left( \sigma' \left( \bar{a}^T \left[ w_F h_f || w_{L \rightarrow F} h_l \right] \right) \right)$$

The similar calculation procedure can be carried out for agent $k$ of class $K$ and agent $l$ of class $L$, the corresponding equations are represented as follows:

$$m_k = \sigma \left( w_K h_k + m_k^K + m_f^K + m_l^K \right)$$

$$m_l = \sigma \left( w_L h_l + m_l^F + m_f^F + m_l^K \right)$$

Besides, the corresponding message representation are computed in a similar manner as abovementioned. At each time step, obtained messages embeddings are passed to other neighbor agents. In this way, one heterogeneous GCN layer can correspond to one round of message passing among neighbor agents and feature updates within each agent. We can extract the high-level message representation of each agent by stacking multiple heterogeneous attention network layers, which correspond to multiple rounds of communication. To stabilize the communication learning process, we extend the multi-head variant of the attentional mechanism to heterogeneous settings and utilizes $M$ heads to obtain features in parallel.
D. Mutual information optimization

In this section, we introduce mutual information optimization to enable more efficient communication learning among agents. For agent \( i \), it obtains message embeddings \( \{m_i^1, m_i^c, \ldots, m_i^n\} \) from neighbor agents of class \((1, c, \ldots, n)\) and fuses them to generate the final message \( m_i \). We define the agent’s immediate neighbor agents as other agents within the field of view of this agent. We utilize the random walk with restart (RWR) [17] to sample a fixed number of samples from the defined neighbors of agent \( i \).

Specifically, the neighbors sampling process are as follows: (1) random walk starting from the agent \( i \), select agents using probability \( p \), and put selected agents to set \( Z_i \). Its total number of agents is fixed, and the number of different types of neighbors are limited to ensure that all types of agents in the initial immediate neighbors are included in the new set \( Z_i \). (2) The agents in set \( Z_i \) are then grouped via types. For agents that belongs to class \( c \), we choose top \( k_c \) agents from set \( Z_i \) according to frequency and collect them as the new set \( N_i(c) \) of \( c \)-class neighbors of agent \( i \).

For agent \( i \), at each timestep, it fuses messages of all \( n \) class \( \{m_i^1, m_i^c, \ldots, m_i^n\} \) to obtain the final message embedding \( m_i \). Nevertheless, some messages of some class may be not useful at a certain time-step. To tackle this issue, we utilize mutual information to implicitly learn which class messages of agent is more valuable at certain time-step, so that the agent can learn the most expressive information from different types of information, so as to better coordinate actions. The mutual information can be calculated by learning a discriminator \( D \) inspired by the idea of [17]. The discriminator \( D \) is aim telling a positive message-value pairs sample \( (\tilde{m}_i^c, Q_i) \) from a negative sample \( (\tilde{m}_i^c, Q_i) \), therefore the corresponding loss function is represented as follows:

\[
L_{MI} = \sum_{i \in I^+} \log D_i(m_i, Q_i) + \sum_{i \in I^-} [1 - \log D_i(\tilde{m}_i, Q_i)] \tag{8}
\]

For agent \( i \), we aim to maximize the MI between the messages \( m_i^c \) of \( c \) class and corresponding individual action value \( Q_i \). The discriminator \( D_i \) is designed to score message-value pairs \( (m_i^c, Q_i) \), we utilize a bilinear layer to be the scoring function which is represented as follows:

\[
D_i(m_i^c, Q_i) = \sigma \left( (m_i^c)^T M_i^c Q_i \right) \tag{9}
\]

where \( M_i^c \) represents a learnable scoring vector, \( \sigma \) utilizes the logistic sigmoid activation function. Therefore, given messages embeddings of \( c \) class and the discriminators, we can maximize the MI utilizing the message embedding-value loss function for \( N \) agents as follows:

\[
L_{MI} = \sum_{i=1}^{I} \sum_{c=1}^{N} L_i^c \tag{10}
\]

\[
L_i^c = \sum_{(c,j) \in N_i^+} \log D_i(m_i^c, Q_i) + \sum_{(c,j) \in N_i^-} \log [1 - D_i(\tilde{m}_i^c, Q_i)] \tag{11}
\]

where \( L_i^c \) represents the message-value loss of \( c \) class, the set \( N_i^+ \) represents a sub-set of set \( Z_i \), in which the agents are sampled from the \( Z_i \) utilizing the RWR. Specifically, a sampling sub-set \( U_i \) is built by selecting agents from set \( Z_i \). For agent \( j \) of the sub-set \( U_i \), if the conditions \( dist(i, j) \leq \delta \) are met, the agent \( j \) and its corresponding class are together added into the sub-set \( N_i^+ \) until the number of sub-set \( N_i^+ \) reaches the batch-size. Where \( dist(\cdot) \) represents the distance of two agents, and \( \delta \) denotes an adjustable parameter that can be set according to different scenarios in the experiment. \( N_i^- \) denotes the complement set of \( N_i^+ \). We utilize the communication learning mechanism designed above to generate the negative message embedding \( \tilde{m}_i^c \) based on the set \( N_i^- \). The designed mutual-information loss function in Eq. (11) can be utilized to maximize the MI.

Except for the proposed MI constraints on the message representations, all the parameters in other modules are generally updated by minimizing the global TD loss. In this paper, CLAR utilizes the mixing network of [4]. Therefore, TD loss function utilized in CLAR is presented as follows:

\[
L_{1D} = \left[ r + \gamma \max_{Q} Q_{\text{tot}} (r', a'; \theta^-) - Q_{\text{tot}}(r, a; \theta) \right]^2 \tag{12}
\]

where \( \theta^- \) represents the parameters of target network, \( \theta \) denotes all parameters in CLAR. Then, the overall optimization of CLAR is presented as follows:

\[
L = L_{TD} + \lambda L_{MI} \tag{13}
\]

where \( \lambda \) represents the adjustable hyper-parameter to achieve a trade-off between the TD loss \( L_{TD} \) and the sum of MI loss of all agents \( L_{MI} \). We set \( \lambda = 0.1 \) for it performs best compared to the other values of \( \lambda \) in the experiments.

IV. EXPERIMENTS

In this section, we select Predator-Capture-Prey (PCP) [16] and StarCraft II Multi-Agent Challenge (SMAC) [18] as our benchmarks. We conduct various experiments on these benchmarks with GPU Nvidia RTX 2080 to answer: Q1: Whether CLAR can improve performance in diverse heterogeneous scenarios? Q2: Whether CLAR can be applied to large-scale multi-agent scenarios? Q3: Does the superiority of CLAR come from communication learning and MI optimization? Q4: How are the learned message embeddings distributed in the representation space and how do they affect team-work and action coordination? More details about experiment setting and algorithm are published to facilitate future research\(^1\).

\[^1\text{https://github.com/Ayeliuk/Clar}\]

![Fig. 3: Illustrations of PCP and SMAC](image-url)
A. Environments and baselines

As shown in Fig. 3, we first select the heterogeneous environment PCP to conduct experiments, which contains three types of agents, predators, captures, and preys. Predators and captures are cooperative, while preys is adversary. Predators can observe environment while the captures cannot obtain any observation from the environment. Therefore, captures necessitate communication and coordination with predators. We build the communication learning module utilizing heterogeneous graph attention networks with MI optimization in the proposed method.

To further demonstrate the effectiveness and superiority of CLAR, we evaluate it on the more complicated benchmark SMAC and select the challenging heterogeneous scenarios of SMAC as shown in Fig. 3. Our experiments are conducted based on the PyMARL framework [18] and utilize its default structure and hyper-parameter settings of the value decomposition module. The hyper-parameters of the proposed communication learning module are set as follows: \( p \) is set to 0.6, \( \delta \) is set to 5. \( Z_i \), \( N_i^- \) and \( N_i^+ \) can be adjusted according to different scenarios. The rest part is set as same as in the PCP environment.

We select 2 value decomposition methods Q Mixing network (QMIX) [4], Q duplex network (QPLEX) [5] and 3 communication learning methods Nearly Decomposable Q network (NDQ) [3], Targeted Multi-Agent Communication (TarMAC) [9], and Heterogeneous Policy Network (HetNet) [16] as baselines, in which TarMAC and HetNet utilize GCN in the communication learning module.

B. Performance

Effectiveness (Q1). Table I shows the average reward of the baselines and CLAR of 3 different random-seed initialization on PCP, in which the bold numbers represents the highest performance results. Fig. 4 shows the average win rate of the baselines and CLAR of 5 different random-seed initialization on SMAC, while the shadow in represent a 95% confidence interval. As shown in Table I and Fig. 4, the proposed method outperforms other MARL baselines on diverse heterogeneous scenarios, which may be due to the effective representation and communication learning of heterogeneous agent features.

Scalability (Q2). To further verify that CLAR can be extended to large-scale heterogeneous scenarios, we compared the performance of CALR and baselines with different numbers of agents. The number of agents varies from 5 to 40, with the predators, captures, and preys ratio being 3:1:1. As shown in Table I, CLAR always performs optimally as the number of agents increases. The results demonstrate that the proposed method CLAR can be extended to large-scale scenarios.

### Table I: Performance of different methods on PCP.

<table>
<thead>
<tr>
<th>Methods</th>
<th>n=5</th>
<th>n=10</th>
<th>n=20</th>
<th>n=40</th>
</tr>
</thead>
<tbody>
<tr>
<td>QMIX</td>
<td>-0.42±0.07</td>
<td>0.38±0.06</td>
<td>-0.35±0.06</td>
<td>-0.30±0.05</td>
</tr>
<tr>
<td>QPLEX</td>
<td>-0.39±0.05</td>
<td>0.35±0.04</td>
<td>-0.29±0.03</td>
<td>-0.26±0.03</td>
</tr>
<tr>
<td>NDQ</td>
<td>-0.34±0.05</td>
<td>0.28±0.04</td>
<td>-0.17±0.04</td>
<td>-0.12±0.03</td>
</tr>
<tr>
<td>TarMAC</td>
<td>-0.30±0.04</td>
<td>0.23±0.04</td>
<td>-0.05±0.02</td>
<td>+0.06±0.02</td>
</tr>
<tr>
<td>HetNet</td>
<td>-0.31±0.05</td>
<td>0.25±0.04</td>
<td>-0.19±0.02</td>
<td>-0.13±0.03</td>
</tr>
<tr>
<td>CLAR</td>
<td>-0.26±0.03</td>
<td>-0.18±0.02</td>
<td>+0.07±0.01</td>
<td>+0.15±0.01</td>
</tr>
</tbody>
</table>

Contributions (Q3). To evaluate the contributions of each component of CLAR, we design an ablation experiments, in which three variants of CLAR are selected as the baselines. As shown in Table II, CLAR-H is the CLAR without heterogeneous GCN. It directly uses the normal GCN for communication learning. CLAR-V is CLAR without the value decomposition component. CLAR-M is the CLAR without MI optimization. The performance of all three variants decreases compared to the CLAR, illustrating the effectiveness of each component. The heterogeneous GCN can achieve efficient communication for heterogeneous agents, the MI optimization can further enhance the quality of communication, and the value decomposition can promote the policy learning.

### Table II: Ablation experiments on SMAC.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MMM2</th>
<th>1c3s5z</th>
<th>2c3s5z</th>
<th>1o2r vs 4r</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLAR-H</td>
<td>77.32±6.09</td>
<td>89.32±4.16</td>
<td>82.13±6.43</td>
<td>78.49±7.14</td>
</tr>
<tr>
<td>CLAR-M</td>
<td>83.16±5.52</td>
<td>94.17±3.59</td>
<td>87.16±4.60</td>
<td>83.05±5.17</td>
</tr>
<tr>
<td>CLAR-V</td>
<td>85.83±4.75</td>
<td>95.25±2.43</td>
<td>90.61±3.24</td>
<td>87.73±3.67</td>
</tr>
<tr>
<td>CLAR</td>
<td>87.55±4.52</td>
<td>97.58±1.94</td>
<td>93.03±2.17</td>
<td>90.42±3.29</td>
</tr>
</tbody>
</table>

Visualizations (Q4). As shown in Fig. 5, the message representations produced by CLAR-M are almost randomly distributed in the representation space. On the contrary, with the proposed MI optimization, the message representations produced by CLAR automatically cluster several clusters in the space. According to the locations of message representations, we divide the agents into groups. We color the message embeddings by the group to which each agent belongs. That is, for each group in the space, the message embeddings are colored...
uniformly. In the video frame of the same time-step, we can see the correspondence between the agent groups formed in the game and those formed in the message representation space. Agents in the same group tend to receive similar message embedding and accomplish more cooperation.

V. CONCLUSIONS

This paper provides a novel GNN-based communication learning mechanism and MARL framework for heterogeneous scenarios. To our knowledge, our work is the first attempt to solve the heterogeneous multi-agent tasks by integrating heterogeneous GNN, MI optimization and value factorization, which simultaneously solves the issues of scalability, effective communication and action coordination in heterogeneous scenarios. We believe that the proposed method can provide a new way for other researchers to solve the MARL problem.

In the future, further implementation of sub-task partitioning of heterogeneous agents is a promising direction that can be explored to build efficient and scalable heterogeneous multi-agent systems. We intend to apply the proposed method in real-world heterogeneous multi agent scenarios, such as traffic signal control.

REFERENCES

Identifying Influential Spreaders in Complex Networks Using Neighborhood Network Structure

Ziyi Zhang, Rong Yan*, Wei Yuan
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—Identifying influential spreaders is a hot topic in complex network research. While centrality-based algorithms are easy to implement, they often have lower accuracy. Topology-based algorithms are effective for identifying network center influential spreaders but may not perform well in identifying peripheral influential spreaders. To address these problems, we propose a Neighborhood Structure Centrality (NSC) algorithm, which utilizes structural embedding and clustering to collect various network structural information and calculates node influence based on both node and neighborhood structural information. We compare the NSC algorithm with twelve baseline algorithms on six public datasets and four synthetic network datasets and demonstrate its higher accuracy.

Index Terms—complex network, influential spreaders identification, network structure.

I. INTRODUCTION

At present, researchers have conducted extensive studies on evaluating node influence, which can be mainly divided into two categories according to application scenarios: identifying influential spreaders and influence maximization. This paper focuses on identifying influential spreaders. We evaluated the influence of nodes by analyzing network structure and node characteristics, and ranked nodes according to their influence values, considering the top-ranked nodes as influential spreaders. In recent years, identifying influential spreaders has received extensive attention [1, 2]. The following provides an overview of the current state of research on algorithms for identifying influential spreaders.

In general, identifying influential spreaders algorithms can be roughly divided into two categories, including centrality-based algorithm and topology-based algorithm. Centrality-based algorithm also includes local centrality-based algorithms (LCAs) [3, 4], global centrality-based algorithms (GCAs) [5, 6], and semi-global centrality-based algorithms (SCAs) [7–9]. In general, centrality-based algorithms solely considering the features of nodes and neighbors will lead to ignoring differences in network position and structure, thereby affecting the accuracy and effectiveness of the algorithm. Meanwhile, much works focus on using the network topology to identify the influential spreaders, including k-shell algorithm [10], k-shell-based improved algorithms (KIAs) [11–13], k-shell-based hybrid algorithms (KHAs) [14–16], local global algorithms (LGAs) [17, 18], and structure-based graph neural network algorithms (SGNNs) [19, 20]. Overall, topology-based algorithms ignore high-influence nodes at the periphery of the network, which leads to the influential spreaders aggregation phenomenon. In addition, some hybrid algorithms use multiple topological attributes to evaluate the node influence, but they have to set parameters to adjust the weight of the attributes. Graph neural network algorithms use small network training models to predict all networks, which makes the models cannot learn enough network features. In brief, the recognition results of these algorithms are not stable enough.

In this paper, we identify influential spreaders by network topology. The main contributions of this paper are:

- Presents a new algorithm that uses the neighborhood structure to identify influential spreaders, alleviating the position limitation problem of topological algorithms.
- Taking structural embedding and clustering to collect comprehensive structural information, which alleviates the limited information problem of centrality-based algorithms and improves their accuracy.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 introduces our proposed algorithm. Section 4 reports the results and the analysis. Finally, in Section 5 we conclude the paper and present future work.

II. RELATED WORK

In this section, we will briefly introduce related work. Firstly, we introduce the definition of the network and the two algorithms used in this paper. Secondly, we summarize several popular identifying influential spreaders algorithms.

A. Basic definition

Let an undirected unweighted network \( G(V, E) \) is composed of \( N = |V| \) nodes and \( M = |E| \) edges, where \( V = \{v_1, v_2, \ldots, v_n\} \) represents the set of nodes in the network and \( E = \{e_1, e_2, \ldots, e_m\} \) represents the set of edges.

In this paper, we use the GraphWave and KMeans++ algorithms. GraphWave [21] is a structural embedding algorithm.
that utilizes spectral analysis to convert a network into a signal and learns node neighborhood structural features through spectral analysis techniques. This algorithm can capture nodes with similar local structures throughout the network, and generate node representations using these structures. KMeans++ [22] is a clustering algorithm that can partition nodes into k pre-specified categories. Nodes within each category have high similarity, while the similarity between nodes from different categories is low.

B. Identifying influential spreaders algorithms

Kitsak et al. [10] proposed the k-shell algorithm, which categorizes nodes into different shells to evaluate their influence. The advantage of the k-shell algorithm is its high efficiency and good recognition results. However, its disadvantages include position limitations and monotonicity problems, meaning that it cannot accurately evaluate the influence of nodes located at the network periphery and cannot distinguish the influence of nodes within the same shell. In recent years, many efforts have been devoted to addressing these problems, such as NC+ [12], G+ [13], KSH [15], and WKSD [16].

NC+ iteratively calculates the sum of neighbors’ k-shell indexes to evaluate node influence, which improves the accuracy of the algorithm but exacerbates the position limitation problem. G+ refers to the gravity formula of physics, which regards the k-shell index value of the node as the mass, and the path length from the node to the neighbor as the distance to calculate the influence of the node. G+ achieves higher accuracy but still has a position limitation problem. KSH evaluates the influence of nodes based on k-shell power, degree, and distance. While it alleviates the position limitation problem, it sets a tunable parameter that may lead to unstable results. WKSD evaluates node influence using the k-shell index and degree. It is effective on incompletely connected networks, but the inclusion of two tunable parameters may lead to unstable results.

In addition to the related algorithms of k-shell, neural networks are also used to identify influential spreaders. RCNN [19] extracts each node’s neighborhood subnetwork and constructs a micro-feature matrix, then it uses CNN to predict the influential spreaders. Based on RCNN, M-RCNN [20] adds the community attributes and macro attributes to construct a three-channel node feature matrix, and automatically learns the weights of the three attributes to predict the influential spreaders. Although these two algorithms can automatically adjust parameters to address the instability problem, their accuracy is relatively low when applied to large-scale networks.

III. OUR ALGORITHM

In this section, we introduce our proposed NSC algorithm in detail. Firstly, we give some basic notation definitions as shown in Table I. NSC algorithm includes three stages: (1) structure embedding, (2) clustering and cluster index allocation, and (3) node influence calculation.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_i$</td>
<td>Degree of node $v_i$, $v_i \in V$ in $G$</td>
</tr>
<tr>
<td>$k$</td>
<td>Number of clusters</td>
</tr>
<tr>
<td>$d_{uv}$</td>
<td>The distance between nodes $u$ and $v$</td>
</tr>
<tr>
<td>$KC_i$</td>
<td>The $i$-th cluster</td>
</tr>
<tr>
<td>$neigh_{inf}(v)$</td>
<td>The influence of node $v$ neighbors</td>
</tr>
<tr>
<td>node_{inf}(v)</td>
<td>The influence of node $v$</td>
</tr>
</tbody>
</table>

A. First stage: structure embedding

In this stage, we try to capture the topology information of the network by using GraphWave algorithm [21]. GraphWave algorithm uses low-dimensional embedding to represent the node, where nodes with similar neighborhood structures have similar embeddings. Generally, GraphWave algorithm is divided into three steps: (1) spectral graph wavelet diffusion, (2) node mapping, and (3) sampling. Once the network structure embedding is completed, nodes with similar structures are closer, while nodes with different structures are farther apart from each other.

B. Second stage: clustering and cluster index allocation

In this stage, we designed a clustering index to distinguish structural importance, and to improve the efficiency of the algorithm, we chose the KMeans++ algorithm [22] for clustering. In this paper, we assume that the node influence within each cluster is similar. Firstly, we obtain $k$ clusters by clustering, each cluster corresponds to one structure and ultimately captures the network of $k$ structures. Next, we use the cluster index to represent the node importance within the cluster. In large-scale networks, the embedding vectors of nodes clustering boundaries are not apparent, which can cause incorrect clustering results for some nodes. Therefore, it is necessary to fine-tune the clustering. We first calculate the average degree $avg(C_i)$ of each cluster $C_i$, and sort all clusters $C = \{C_1, C_2, \ldots, C_k\}$ in ascending order by $avg(C_i)$. Then, the nodes whose degree is greater than $avg(C_i)$ in the cluster $C_i$ are exchanged with the nodes whose degree is less than $avg(C_{i+1})$ in the set $C_{i+1}$. The number of exchange nodes is $30\%$ of the number of nodes to be adjusted in smaller clusters. After this processing, we recalculate $avg(C_i)$ and sort $C$ in ascending order, and give the index of cluster $C_i$ as $KC_i$, the larger $avg(C_i)$ the higher the value of $KC_i$, $KC_i \in [1, k]$.

Algorithm 1 describes the details of this stage. Where the input is the embedding vector $X$ of nodes and the number of clusters $k$. Line 1 is to cluster $X$. Lines 2-4 are to calculate the average degree of each cluster, lines 5-7 are to fine-tune the nodes in the clusters and sort the clusters. Lines 8-13 are to assign cluster indexes to each cluster.

C. Third stage: node influence calculation

In this stage, we calculate the node influence and then sort the nodes according to their influence within the network. In the experiment, we find that the clustering stage will produce deviation in large-scale networks or high-density networks.
Algorithm 1: Clustering and cluster index allocation

Input: Structural embedding X, Number of clusters k.
Output: Rank[C, Scorekk].

1. KMeans++ clustering to get a set of k clusters C
2. for each cluster C_i ∈ C do
3.   d_i = average_degree(C_i)
4. end
5. Sort C in ascending order by d_i
6. Fine-tuning the nodes in the cluster
7. Recalculate d_i and resort C in ascending order by d_i
8. Set score = 1
9. for each cluster C_i ∈ C do
10.   The index value of the i-th cluster KC_i = score
11.   Append C_i and KC_i to Rank[C, Scorekk]
12. score++ = 1
13. end

Network graph

1. Calculate KC and neigh_inf(v)
2. Fine-tune the nodes of each cluster
3. Calculate node_inf(v)
4. Calculate KC and neigh_inf(v)
5. Sort by average degree among clusters, sort by node_inf(v) within clusters
6. Sort the nodes by node_inf(v)

Obtain influential spreaders list

Fig. 1: The procedure of node influence calculation.

Thus, we design different node influence calculation strategies according to the network size. On the one hand, for small networks, we sort the nodes within each cluster first and then sort the clusters based on their influence. On the other hand, for larger networks, we directly sort all nodes according to their influence. Figure 1 shows the procedure of node influence calculation. Node influence includes the influence of the node itself as well as the influence of its neighbors.

For networks with less than 10,000 nodes, we consider node influence as a combination of its degree and the influence of its nearest neighbors. Represent neighborhood influence using the cluster index value of neighbors. If the cluster index value of a node is high, it indicates that the local structure where it is located has a significant influence. Then, the influence of node v and the influence of node v’s neighbors are defined in Eq.(1) and Eq.(2) respectively.

\[ \text{node}_{-}\text{inf}_1(v) = d_v + \text{neigh}_{-}\text{inf}_1(v) \]  \hspace{1cm} (1)

\[ \text{neigh}_{-}\text{inf}_1(v) = \sum_{j \in \eta(v)} \text{KC}_j \]  \hspace{1cm} (2)

where \( \eta(v) \) represents all nodes within the nearest neighborhood of node \( v \), \( d_v \) is the degree of node \( v \), \( \text{KC}_j \) is the cluster index value of node \( j \).

Algorithm 2: Node influence calculation

Input: Network graph \( G(V, E) \), Rank[C, Scorekk].
Output: Rank[V].

1. if Node number |V| < 10,000 then
2.   for each nodes \( v_i \in V \) do
3.     \( \text{node}_{-}\text{inf}_f(v_i) = 0 \)
4.     for \( v_i \in \text{neighbors}(v, r) \) do
5.       \( \text{neigh}_{-}\text{inf}_f(v_i) \) += \( \frac{\text{KC}(w) \cdot d_j}{L_{v_j}} \)
6.   end
7. else
8.   for each nodes \( v_i \in V \) do
9.     \( \text{node}_{-}\text{inf}_f(v_i) = 0 \)
10.    for \( v_i \in \text{neighbors}(v, r) \) do
11.       \( \text{node}_{-}\text{inf}_f(v_i) \) += \( \text{neigh}_{-}\text{inf}_f(v_i) \)
12. end
13. end

14. for each nodes \( v_i \in V \) do
15.   \( \text{node}_{-}\text{inf}_f(v_i) = \sum_{w \in \eta(v)} \text{neigh}_{-}\text{inf}_f(w) \)  \hspace{1cm} (3)
16.   \( \text{neigh}_{-}\text{inf}_f(w) = \sum_{j \in \eta(w)} \frac{\text{KC}_w \cdot d_j}{L_{w_j}^2} \)  \hspace{1cm} (4)

17. end

18. Rank[V] = Resorting all nodes by node_inf2(v_i).

For the network with more than 10,000 nodes, we define the influence of node \( v \) and the influence of node \( v \)’s neighbors as shown in Eq.(3) and Eq.(4) respectively.

\[ \text{node}_{-}\text{inf}_f(v) = \sum_{w \in \eta(v)} \text{neigh}_{-}\text{inf}_f(w) \]  \hspace{1cm} (3)

\[ \text{neigh}_{-}\text{inf}_f(w) = \sum_{j \in \eta(w)} \frac{\text{KC}_w \cdot d_j}{L_{w_j}^2} \]  \hspace{1cm} (4)

where \( L_{v_j} \) is the distance from node \( v \) to node \( j \), \( \eta_r(v) \) denotes the set of all nodes within \( r \) hops of node \( v \), and \( \eta(v) \) denotes the nearest neighbor node set of node \( v \). In the experiment, we set \( r = 3 \).

Algorithm 2 describes the details of this stage. The input of the algorithm is the network \( G \) and the clusters with their index, and the output is the influence sequence. Lines 1-12 compute the node influence for networks with less than 10,000 nodes, while lines 13-29 compute the node influence for networks with more than 10,000 nodes. The calculation process for both parts is similar. Lines 1-3 and 14-19 compute the self-influence of nodes, while lines 4-7 and 20-25 compute the influence of the neighbors of the nodes. Lines 8-11 and 26 sort the nodes.

D. Computing complexity

Given a network with \( N \) nodes, \( E \) edges, \( R \) neighbors within 3 hops of each node, \( k \) clusters, and 2-dimensional node embedding vectors. The time complexity of the network embedding phase is \( O(S|E|) \), where \( S \) is the order Chebyshev polynomial approximation. The time complexity of the clustering and cluster index allocation phases is \( O(2Nk + k) \), and the node influence calculation and sorting phases are
Overall, the total time complexity of the NSC algorithm is $O(S|E| + 2Nk + k + NR + N\log N)$.

IV. Experiments

We conduct experiments on six real-world networks and four synthetic networks. In the experiment, we compare twelve baseline algorithms to verify the effectiveness of our proposed NSC.

A. Baselines

Degree centrality (DC) [3]: It evaluates the influence of a node by using the number of its nearest neighbors. Betweenness centrality (BC) [6]: It uses the number of shortest paths through the node to evaluate node influence. Eigenvector centrality (EC) [7]: It evaluates the influence of a node based on the importance of its neighbors. K-shell (KS) [10]: It evaluates the influence of a node based on its position in the network. PageRank [8]: It evaluates the influence of a node based on the probability of the node being visited during random walks. Ksum [23]: It evaluates the node influence by calculating the sum of the neighbor’s degree. Neighborhood coreness centrality (NC+) [12]: It evaluates the influence of nodes by calculating the sum of the neighbor’s k-shell. Gravity index centrality (G+) [13]: It refers to the gravity formula, where the k-shell of nodes is taken as the mass, and the shortest path length between nodes is taken as the distance to evaluate node influence. Global and local structure (GLS) [17]: It evaluates the influence of nodes by combining local information with global information. Neighborhood entropy centrality (NEC) [9]: It evaluates the node influence by calculating the sum of the entropies of the node’s neighbors. RCNN [19]: It evaluates node influence by generating a neighborhood feature matrix for each node and applying a convolutional neural network. Multi-channel RCNN (M-RCNN) [20]: It evaluates node influence by generating three-channel node representations and using convolutional neural networks.

B. Dataset

We use six real-world networks and four synthetic networks for the experiments, including Jazz [24], USAir [25], Hamster[26], Power [27], PGP [28], Sex [29], and BA [30]. BA network synthesized by Python’s networkx library. Table II gives the detailed description of each network ($N$ denotes the number of nodes, $M$ denotes the number of edges, $\beta_{hk}$ and $\beta$ denote the infection threshold and infection rate of the network respectively. $D$ is the diameter of the network, $<K>$ is the average degree of the network, and $k$ is the number of clusters in NSC).

C. Evaluation criteria

1) Kendall correlation coefficient $\tau$ [31]: $\tau \in [-1,1]$ is widely used to measure the correlation between two lists. After calculating the influence of each node using the SIR model [32], the accuracy of the algorithm is evaluated by comparing the ranking produced by the algorithm with the actual ranking generated by the SIR model. $\tau$ is mathematically defined as Eq.(5):

$$\tau = \frac{N_c - N_d}{N(N-1)/2}$$

where $N_c$ and $N_d$ denote the number of consistent and inconsistent pairs respectively, and $N$ is the number of nodes in the list. Generally, if the value of $\tau$ is bigger, the algorithm considers effective.

2) Improvement percentage $\eta(\%)$ [33, 34]: To verify the improvement of NSC over the baseline algorithms, we conduct experiments using the $\eta$ function (improve $\tau$ ratio), and $\eta$ is mathematically defined as Eq.(6)

$$\eta(\%) = \begin{cases} \frac{\tau_{C(0)} - \tau_{C}}{\tau_{C}} \times 100, & \tau_{C} > 0 \\ \frac{\tau_{C(0)} - \tau_{C}}{\tau_{C}} \times 100, & \tau_{C} < 0 \\ 0, & \tau_{C} = 0 \end{cases}$$

where $\tau_{C(0)}$ represents the $\tau$ value of NSC, and $\tau_{C}$ represents the $\tau$ value of other algorithms. $\eta(\%) > 0$ means that the algorithm has been improved, $\eta(\%) < 0$ means that NSC is worse than other algorithms, and $\eta(\%) = 0$ means that NSC is not improved compared with other algorithms.

D. Experimental setup

In the process of using the SIR model to obtain the actual influence of nodes, 1,000 simulations were run on networks with less than 10,000 nodes, and 100 simulations were run on networks with more than 10,000 nodes. For the different structures of each network, we set the appropriate number of clusters $k$ for NSC, and the specific values are shown in Table 2. The parameter $L$ for the M-RCNN algorithm is set to 4 in the Power network and 28 in the other networks. The parameter $L$ of RCNN is set to 28 in all networks. RCNN and M-RCNN use BA networks for training, so these two algorithms are not examined in the effectiveness experiment on synthetic networks.

E. Results and analysis

1) Effectiveness experiment on real-world network: We compared our proposed NSC with the twelve baseline algorithms for $\tau$ values, as shown in Fig. 2. It can be seen that NSC can achieve high correlation in most networks. Compared to the two structure-based neural network algorithms, the NSC performance was also more significant. Specifically, for Jazz,
USAir, Hamster, and Sex networks, NSC achieves the highest $\tau$ value. For larger networks like PGP and Sex, the value of $\tau$ for NSC is significantly higher than most algorithms.

Table III shows the average $\tau$ value for each algorithm. In Jazz, NSC achieves the highest $\tau$ value of 0.9143. It is 1.21% better than the second-placed NC+ and 40.3% better than the worst BC. In USAir, the NSC is 0.5% higher than the second-placed G+ and 36.2% higher than the worst BC. In Hamster, the $\tau = 0.9108$ of the NSC is the highest. It is 0.19% better than the second-placed G+ and 40.3% better than the worst PageRank. In Power, the NSC is 6.81% lower than the first-place GLS and 1.91% higher than the third-place NC+.

In conclusion, NSC outperforms other baselines. It shows that the combination of topology and centrality algorithm is effective, and it can obtain more accurate node influence ranking.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Network</th>
<th>Jazz</th>
<th>USAir</th>
<th>Hamster</th>
<th>Power</th>
<th>Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau(DC)$</td>
<td>BA1</td>
<td>0.9198</td>
<td>0.7624</td>
<td>0.7636</td>
<td>0.7244</td>
<td>0.4906</td>
</tr>
<tr>
<td>$\tau(BC)$</td>
<td>BA2</td>
<td>0.5112</td>
<td>0.5293</td>
<td>0.5143</td>
<td>0.5373</td>
<td>0.2915</td>
</tr>
<tr>
<td>$\tau(EC)$</td>
<td>BA3</td>
<td>0.7994</td>
<td>0.7879</td>
<td>0.6163</td>
<td>0.5113</td>
<td>0.5829</td>
</tr>
<tr>
<td>$\tau(\text{PageRank})$</td>
<td>BA4</td>
<td>0.1971</td>
<td>0.2855</td>
<td>0.2559</td>
<td>0.2856</td>
<td>0.2855</td>
</tr>
<tr>
<td>$\tau(KS)$</td>
<td>GLS</td>
<td>0.5333</td>
<td>0.7411</td>
<td>0.7480</td>
<td>0.4826</td>
<td>0.4242</td>
</tr>
<tr>
<td>$\tau(GLS)$</td>
<td>NEC</td>
<td>0.8896</td>
<td>0.8425</td>
<td>0.7746</td>
<td>0.8157</td>
<td>0.8496</td>
</tr>
<tr>
<td>$\tau(\text{PageRank})$</td>
<td>M-RCNN</td>
<td>0.8329</td>
<td>0.6520</td>
<td>0.5923</td>
<td>0.5704</td>
<td>0.3256</td>
</tr>
<tr>
<td>$\tau(\text{PageRank})$</td>
<td>RCNN</td>
<td>0.8345</td>
<td>0.8655</td>
<td>0.8009</td>
<td>0.6933</td>
<td>0.5705</td>
</tr>
<tr>
<td>$\tau(\text{PageRank})$</td>
<td>NSC</td>
<td>0.8310</td>
<td>0.8785</td>
<td>0.8779</td>
<td>0.7267</td>
<td>0.5425</td>
</tr>
<tr>
<td>$\tau(\text{PageRank})$</td>
<td>NEC</td>
<td>0.9022</td>
<td>0.8713</td>
<td>0.8790</td>
<td>0.7285</td>
<td>0.6957</td>
</tr>
<tr>
<td>$\tau(\text{PageRank})$</td>
<td>M-RCNN</td>
<td>0.8423</td>
<td>0.8724</td>
<td>0.8657</td>
<td>0.4147</td>
<td>0.5584</td>
</tr>
<tr>
<td>$\tau(\text{PageRank})$</td>
<td>NSC</td>
<td>0.9145</td>
<td>0.8916</td>
<td>0.9108</td>
<td>0.7476</td>
<td>0.6959</td>
</tr>
</tbody>
</table>

2) Cluster number $k$ analysis: In this experiment, we explore the impact of a different number of clusters $k$ on the $\tau$ value of NSC. Also, we provide guidance on choosing the value of $k$ for different networks, as shown in Fig. 3.

Except for Power, we can observe that NSC achieves the highest average $\tau$ value when $k$ is around 6 in most networks. On a small network like Jazz, the average $\tau$ reaches the highest value when $k$ is 5. On medium-sized networks such as Hamster, NSC achieves the best results when $k$ is 6. For large networks such as PGP, the best results can be achieved when $k \geq 7$. Power is different from other networks, the average $\tau$ value of the NSC decreases with the increase of the $k$ value. This is because all nodes in the Power network have a similar neighborhood structure. Therefore dividing the nodes into more clusters, which will degrade the performance of the algorithm. In summary, we recommend that the value of $k$ should be set between 5 and 13. For small-scale networks set a smaller value of $k$ and for large-scale or dense networks set a larger value of $k$. When the network structure is similar, $k$ is set to 2.

3) Improvement percentage $\eta(\%)$: The experimental results are shown in Fig. 4. In most networks, the NSC algorithm has obvious improvement over other algorithms. The percent improvement $\eta(\%)$ for BC, PageRank, and NEC algorithms is the highest. In Jazz, USAir, Hamster, PGP, and Sex networks, the value of $\eta(\%)$ is greater than zero when $\beta > 0.06$. Especially in Sex, $\tau_{\text{NSC}}$ is relative to $\tau_{\text{PR}}, \tau_{\text{NEC}}, \tau_{\text{BC}}, \tau_{\text{DC}}$ and $\tau_{\text{KS}}$.

Fig. 3: $\tau$ comparisons on different cluster $k$ of NSC.
exceeds 50%.

4) Effectiveness experiment on synthetic network: To further verify the effectiveness of NSC, we conduct experiments on BA networks. Since the nodes in the BA networks have at least \(< K >\) edges, the shell values of all nodes after k-shell decomposition are the same, and the influence of nodes cannot be distinguished, so k-shell is not considered in this experiment. To verify the robustness of NSC on network scale and density, we set different node numbers and average degrees for the network. Table IV shows the average \(\tau\) value of each algorithm in different scale BA networks. We observe that the NSC average \(\tau\) on \(BA_1\) and \(BA_3\) is 0.4% lower than G+, but higher than other algorithms. In other networks, NSC outperforms other algorithms.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose an influential spreaders identification algorithm NSC. We compare NSC with 12 state-of-the-art baseline algorithms on 6 real networks and 4 synthetic networks, the results show that our proposed algorithm can obtain higher \(\tau\) values than other algorithms in most networks. Specifically, compared with the 10 centrality algorithms, our algorithm is more stable when the network size and \(\beta\) value change. Meanwhile, compared with the 2 neural network algorithms based on neighborhood structure, NSC complies higher recognition accuracy, especially in larger networks. In future work, we will optimize the algorithm to automatically cluster based on network size, clustering coefficient, and structural difference metrics, so as to make it more widely applicable.

ACKNOWLEDGMENT

This research is supported by the National Natural Science Foundation of China (Grant No. 61866029) and the Self-project Program of Engineering Research Center of Ecological Big Data, Ministry of Education.

REFERENCES

Session OAEP: Optimization Approaches for Enhanced Performances
Altering Backward Pass Gradients to Improve Convergence

Bishshoy Das*, Milton Mondal*, Brejesh Lall, Shiv Dutt Joshi, and Sumantra Dutta Roy
Indian Institute of Technology Delhi, Hauz Khas, New Delhi - 110016, India
{bishshoy.das, milton.mondal, brejesh, adjoshi, sumantra}@ee.iitd.ac.in

Abstract

In standard neural network training, the gradients in the backward pass are determined by the forward pass. As a result, the two stages are coupled. This is how most neural networks are trained hitherto. Gradient modification in the backward pass has seldom been studied in the literature. In this paper we explore decoupled training, where we alter the gradients in the backward pass. We propose a simple yet powerful method called PowerGrad Transform (PGT), that alters the gradients before the weight update in the backward pass and significantly enhances the predictive performance of a convolutional neural network. PGT trains networks to arrive at a better optima at convergence. It is computationally efficient, and adds no additional cost to either memory or compute, but results in improved final accuracies on both the training and test datasets. PowerGrad Transform is easy to integrate into existing training routines, requiring just a few lines of code. With decoupled training, our method improves baseline accuracies for ResNet-50 by 0.73%, for SE-ResNet-50 by 0.66% and by more than 1.0% for the non-normalized ResNet-18 network on the ImageNet classification task.

1. Introduction

Backpropagation is traditionally used for training deep neural networks [8]. Gradients are computed using basic calculus principles to adjust the weights during backpropagation [7]. However, decoupling the backward pass from the forward pass by modifying the gradients to improve training efficiency and final convergent accuracy has hardly been explored. In this paper we explore the landscape of decoupling the forward pass from the backward pass by altering the gradients and subsequently updating the network’s parameters with the modified gradients. There are several ways to achieve gradient modification in the backward pass. We discuss a few techniques in Fig. 1.

Type 0: No modification: In this method, we calculate the gradients using the standard calculus rules and use the chain rule to calculate the gradients of the rest of the network’s parameters, also known as backpropagation as portrayed in Fig. 1(a). The network is then updated with the gradient descent equation:

\[ W_{t+1}^i = W_t^i - \lambda \nabla W_t^i(L) \quad i = D, D - 1, \ldots, 1 \quad (1) \]

Type I: Independent gradient modification at multiple points: Here the gradients are first computed using standard procedure and then individually altered as shown in Fig. 1(b). Gradient clipping [1] and Adaptive gradient clipping [11] are examples of such modifications. It can be described as:

\[ W_{t+1}^i = W_t^i - \lambda f(\nabla W_t^i(L)) \quad i = D, D - 1, \ldots, 1 \quad (2) \]

where the gradients \( \nabla W_t^i(L) \) are transformed using the transformation function ‘\( f \)’ before the weight update.

Type II: Gradient modification at a point very early in the backward graph: In this type of modification, the gradient is altered at a very early stage in the backward computation graph and then all subsequent gradients are generated using the values obtained with the modified gradients (Fig. 1(c)). Because of the chain rule, network parameters whose gradients are connected to the point of alteration in the computation graph also gets subsequently altered. It can be described as:

\[ W_{t+1}^i = W_t^i - \lambda f(\nabla W_t^i(L)) \quad i = D, D - 1, \ldots, 1 \quad (2) \]

\[ W_{t+1}^i = W_t^i - \lambda \nabla W_t^i(L)^* \quad i = D, D - 1, \ldots, 1 \quad (4) \]

where the gradient \( \nabla W_t^i(L)^* \) is first transformed using the transformation function ‘\( f \)’ and then this transformed gradient is propagated through the rest of the backward graph. All other gradient vectors \( \nabla W_t^i(L)^* \) are computed as it is, but because of the early injection of the transformed gradient \( \nabla W_t^i(L) \), all other gradient vectors that are connected to the transformed gradient through the chain rule \( (\nabla W_t^i(L)^*, i=D-1,\ldots,1) \), gets subsequently altered.

Type I is computationally more expensive than type II as it requires altering the gradients of each and every parameter individually. We propose PowerGrad Transform (PGT), a type II method that modifies the gradients at the softmax layer. The following are the major contributions of this paper:
1. We propose PowerGrad Transform (PGT), which decouples the backward and the forward passes of neural network training. PGT alters the gradients in the backward pass before the update step leading to accelerated training and a significant boost in the network’s predictive performance.

2. We perform theoretical analysis of the properties of the PGT and also show that in non-BN networks, PGT can be used to increase the network’s convergence rate and improve the final accuracy.

3. We find that PGT improves the performance for a variety of models (non-BN and BN ResNets, SE-ResNets) using the ImageNet dataset. We empirically conclude that PGT helps a network to improve by locating a more optimum convergence point.

2. Related Works

Gradient Clipping [11] is a gradient modification method that involves clipping/altering the gradients with respect to a predefined threshold value during backward propagation through the network and updating the weights using the clipped gradients [13]. By rescaling the gradients, the weight updates are likewise rescaled, significantly reducing the risk of an overflow or underflow [10]. GC can be used for training networks without batch-normalization. At larger batch sizes, the clipping threshold in GC becomes highly sensitive and requires extensive finetuning for various models, batch sizes, and learning rates. Adaptive Gradient Clipping [11] is developed to further enhance backward pass gradients than what is performed by GC. It takes into account the fact that the ratio of the gradient norm to the weight norm can provide an indication of the expected change in a single step of optimization. Label smoothing, introduced by Szegedy et al. [14], utilizes smoothing of the ground truth labels as a method to impose regularization on the logits and the weights of the neural network. AGC performs better than GC in non-normalized networks. However, we show that PGT outperforms both in networks such as ResNets.

Knowledge distillation (KD) [5] is a process in which two networks are trained with hard and soft labels alternatively. Variants of knowledge distillation include self-distillation [16], channel distillation [3]. Even though both PGT and KD require probability manipulation, the key difference is that in the latter the transformation is applied in the forward pass, while PGT is a backward pass modification only. PGT differs from self-knowledge distillation as it neither introduces any additional sub-modules nor creates different ensembles to improve the performance of the model. PGT follows the standard neural network training mechanism with modified gradients.

3. PowerGrad Transform

A neural network with parameters \( W \) generates \( C \) logits denoted by \( z \) for every input vector \( x \). \( z \) is given as \( z = Wx \). Then a set of probability values \( p_i \) are generated from the logits using a softmax function which is defined as \( p_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \). \( p_i \) and \( z_i \) represent the predicted probability values and the logits for the \( i^{th} \) class respectively. If the loss function is cross-entropy loss, then the value of the loss is given as \( L = -\sum_{i=1}^{C} q_i \log (p_i) \) where \( q_i \) is the ground truth label of the \( i^{th} \) class for a particular training example. By standard gradient update rule, we can calculate the gradient of the loss with respect to the logits \( \frac{\partial L}{\partial z_i} = p_i - q_i \).

The PowerGrad Transform technique is now described. We introduce a hyperparameter \( \alpha \), which takes a value between \([0, 1]\) and regulates the degree of gradient modification. The PowerGrad Transform method modifies the predicted probability values in the backward pass as follows:

\[
p_i' = \frac{p_i^{\alpha}}{\sum_{j=1}^{C} p_j^{\alpha}} \quad i = 1, \ldots, C \quad 0 \leq \alpha \leq 1 \quad (5)
\]

The above transformation changes the gradient of the loss with respect to the logits as follows:

\[
\frac{\partial L}{\partial z_i} = p_i' - q_i \quad (6)
\]

The rest of the backward pass proceeds as usual.
3.1. Properties of the PowerGrad transformation

We use the same setup as described in section [8]. To explore the properties of PGT, we start by investigating the effect of the transform on the softmax probabilities.

**Lemma 1.** For any arbitrary probability distribution \( P \) with probability values given by \( p_i \) for \( i = 1, \ldots, C \), the corresponding transformed probability values \( p'_i \) given by [Eq. 5] has a threshold \( \left( \sum_{j=1}^{C} p_j \right)^{\frac{1}{\alpha-1}} \) and

\[
p'_i \geq p_i, \text{ if } p_i \leq \left( \sum_{j=1}^{C} p_j \right)^{\frac{1}{\alpha-1}}
\]

\[
p'_i < p_i, \text{ if } p_i > \left( \sum_{j=1}^{C} p_j \right)^{\frac{1}{\alpha-1}} \tag{7}
\]

We call this threshold, the stationary threshold. The stationary threshold is that value of \( p_i \) that does not change after the transformation. However, when \( p_i \) is greater than the stationary threshold, \( p'_i < p_i \).

**Proposition 1.** At \( \alpha = 0 \), the stationary threshold equals \( 1/C \) and all values of the transformed distribution \( p'_i \) reduces to the uniform distribution for \( i = 1, \ldots, C \).

**Proof.** From Eq. (7), we see that the stationary threshold at \( \alpha = 0 \) is \( 1/C \). Also, following from the definition of the transformed probabilities (Eq. [5]) we conclude that if \( \alpha = 0 \), then all values of \( p'_i \) are \( 1/C \). Therefore the transformed distribution at \( \alpha = 0 \) is a uniform distribution.

**Theorem 1.** For any arbitrary probability distribution \( P \) with probability values \( p_i \) for \( i = 1, \ldots, C \), the stationary threshold of the transformed distribution \( P' \) with probability values \( p'_i = \frac{p_i^\alpha}{\sum_{j=1}^{C} p_j^\alpha}, 0 \leq \alpha \leq 1 \) is a monotonically non-decreasing function with respect to \( \alpha \).

**Proof.** To prove monotonicity, we first compute the gradient of the stationary threshold with respect to the variable in concern, \( \alpha \).

\[
\frac{\partial}{\partial \alpha} \left( \sum_{j=1}^{C} p_j^\alpha \right)^{\frac{1}{\alpha-1}} = \frac{1}{\alpha(\alpha-1)^2} \left( \sum_{j=1}^{C} p_j^\alpha \right)^{\frac{1}{\alpha-1}} \times \left( \frac{\sum_{j=1}^{C} p_j^\alpha \log (p_j^\alpha) - \log \left( \sum_{j=1}^{C} p_j^\alpha \right)}{(\alpha - 1)^2} \right) \tag{8}\]

If \( a_1, \ldots, a_n \) and \( b_1, \ldots, b_n \) are non-negative numbers, then using the log sum inequality, we get \( \sum_{j=1}^{n} a_j \log \left( \frac{a_j}{b_j} \right) \geq \left( \sum_{j=1}^{n} a_j \right) \log \left( \frac{\sum_{j=1}^{n} a_j}{\sum_{j=1}^{n} b_j} \right) \).

Setting \( a_j = p_j^\alpha \) and \( b_j = 1 \), we get the following lower bound

\[
\sum_{j=1}^{C} p_j^\alpha \log (p_j^\alpha) \geq \left( \sum_{j=1}^{C} p_j^\alpha \right) \log \left( \frac{1}{C} \sum_{j=1}^{C} p_j^\alpha \right) \tag{9}
\]

Substituting (9) in (8), we get:

\[
\frac{\partial}{\partial \alpha} \left( \sum_{j=1}^{C} p_j^\alpha \right)^{\frac{1}{\alpha-1}} \geq \frac{1}{\alpha(\alpha-1)^2} \left( \sum_{j=1}^{C} p_j^\alpha \right)^{\frac{1}{\alpha-1}} \times \left( (1 - \alpha) \log (C) - \log \left( \sum_{j=1}^{C} p_j^\alpha \right) \right) \tag{10}
\]

\( p^\alpha \) is concave, and so by Jensen’s inequality we get the following upper bound for the second term:

\[
\left( \frac{1}{C} \sum_{j=1}^{C} p_j \right)^{\alpha} \geq \frac{1}{C} \sum_{j=1}^{C} p_j^\alpha \tag{11}
\]

\[
\Rightarrow \log \left( \sum_{j=1}^{C} p_j^\alpha \right) \leq (1 - \alpha) \log (C) \tag{12}
\]

Substituting (12) in (10).

\[
\frac{\partial}{\partial \alpha} \left( \sum_{j=1}^{C} p_j^\alpha \right)^{\frac{1}{\alpha-1}} \geq 0 \tag{13}
\]

We conclude that the stationary threshold is a monotonic non-decreasing function with respect to \( \alpha \). Also the derivative of PGT function with respect to the true probabilities is non-negative which in turn means that the transformation is an order-preserving map. All values greater than the threshold move towards the threshold after transformation and all values below the threshold also move towards the threshold, and the threshold itself moves monotonically towards \( 1/C \) as \( \alpha \) is decreased from 1 to 0. This concludes that the transformation smooths the original distribution.

4. Experiments

We perform experiments on different variants ResNets using the ImageNet-1K dataset [2]. All models are trained on four V100 GPUs with a batch size of 1024. We utilize a common set of hyperparameters for all experiments, which are as follows: 100 epoch budget, 5 epochs linear...
warmup phase beginning with a learning rate of $4 \times 10^{-4}$ and ending with a peak learning rate of 0.4, a momentum of 0.9 and weight decay of $5 \times 10^{-4}$, the SGD Nesterov optimizer and mixed precision. In our studies, we employ either a step scheduler (dividing the learning rate by 10 at the 30th, 60th, and 90th epochs) or a cosine decay scheduler [9]. We find $\alpha = 0.25$ and $\alpha = 0.05$ to be good choices for ResNet-18 and ResNet-50, though larger values such as $\alpha = 0.3$ also have good performance as well. We use $\alpha = 0.3$ for Squeeze-and-Excitation variant of ResNet-50 i.e. SE-ResNet-50 [6]. The experimental results are shown in Table 1.

With consistent improvements in training and test accuracies across all cases, we conclude PGT helps networks learn better representations and arrive at better optimas during convergence. Per epoch training and test accuracy plots of ResNet-18 and ResNet-50 (both with and without PGT) are shown in Fig. 2(a,b). Practioners can also choose to accelerate training and save as much as 40% of the epoch budget [Fig. 2(c)].

### 4.1. Empirical studies on networks without Batch Normalization

We examine issues that occur in non-normalized networks (networks without BN layers). We use ResNet-18 [4] as the foundation model trained on ImageNet-1K [12]. Deeper networks such as ResNet-34 and ResNet-50 are impossible to train without Batch Normalization courtesy of the increased depth and so we solely focus on ResNet-18. Throughout the training process, we monitor variations in the the per-filter L2-norm of each layer’s weights. In Fig. 3(a), some filters of layer 11 achieve a norm of zero during training. We refer to this event as ‘Zeroing Out’ (Fig. 4), and it occurs when one of the channels (or filters) of a weight tensor gets fully filled with zeros and such filters do not contribute at all to determine the input-output relationship of a dataset, as the feature tensor it produces is also filled with zeros for the corresponding filter. When a filter once zeroes out, it does not recover with further training, as all gradients that it receives in future iterations are all zeros. Fig. 3(c) is the plot of the final conv layer’s filters (layer 19) and the features output of final global average pooling (GAP) layer respectively. We observe a number of filters and features in the final layer completely zeroing out. Gradient modification methods such as PGT can alleviate the zeroing out phenomena as we observe that the number of zeroed out filters has considerably reduced [Fig. 3(b, d)]. The feature vector after the GAP layer [Fig. 3(c)] directly interfaces with the fully connected layer. Therefore any zeroed out features leads to permanent information loss, as it does not contribute to the learning of decision boundaries in the fully connected layer. Also, zeroed out weights ten-

![Figure 2: Log-log plots of training and test accuracies and comparison with baseline of batch-normalized variants:](a) ResNet-18 ($\alpha = 0.25$), (b) ResNet-50 ($\alpha = 0.3$). (c) Training speed comparison between PGT (60 epochs) and baseline (100 epochs). They both converge to the same test accuracy (76.5%) on ImageNet-1K. PGT’s accelerated training saves 40% of the epoch budget.

![Table 1: Results and comparison table for networks trained on ImageNet-1K. Best training and test accuracies are highlighted in red and blue respectively. Accuracy differences are highlighted in yellow.](https://github.com/skalien/power-grad-transform)
sors lead to zeroed out gradients hence stopping training for all subsequent iterations leading to a collapse in training for the affected layers. With large batch sizes, it is possible that an entire layer zeroes out as shown in the figure below. The final feature tensor [Fig. 3(d)] with PGT enabled, does not contain any zeroed out regions indicating that information loss is mitigated as the features pass on from the feature extracting layers to the fully connected layer.

In Table 2 we find that the baseline performance for high batch sizes (1024) is drastically inferior to baselines for other batch sizes. PGT helps regain some of the lost performance by 0.682% (65.498% vs. 64.816%). At a batch size of 512, invoking PGT improves the training accuracy baseline by 1.48% and the test accuracy baseline by 0.684%, while at a batch size of 256, the improvement in training and test accuracies are 1.11% and 1.018% respectively. In comparison, the test accuracy improvements obtained by GC and AGC at batch size of 256 is much less at 0.27% and 0.5%, respectively. On the training accuracy front, since we get a significant boost (1.48% at batch size of 512 and 1.11% at batch size of 256), it leads us to infer that when PowerGrad Transform is used, the network fits the training dataset more tightly and the convergence optimum is significantly superior. When AGC and PGT are combined, we see a tremendous increase in test accuracy of over 2.06% over the baseline.

5. Ablation Study

We conduct ablation studies to investigate the effects of PowerGrad Transform for different values of the hyperparameter ($\alpha$), where we use the ResNet-50 architecture and combine our proposed method with different schedulers, regularization techniques and different values of $\alpha$. We report our findings in Table 3. First we examine the effect of PGT on the step scheduler baseline in order to later compare it to the cosine scheduler baseline. **Row-1** We begin with the step scheduler baseline (75.97%). **Row-2** PGT improves upon the step scheduler baseline (test set) by a substantial margin with 0.524% (76.494% as opposed to 75.97%). **Row-3** Introducing the cosine scheduler yields a 0.59% improvement (76.56% vs. 75.97%) over the step scheduler. **Row-4** After introducing label smoothing, the test accuracy relative to the cosine scheduler baseline increases by only 0.138% (from 76.56% to 76.698%). **Row-5** However, introducing PGT with $\alpha = 0.3$ alone (without label smoothing) improves the cosine scheduler baseline by 0.326% (76.886% vs. 76.56%). **Row-6** Combining PGT ($\alpha = 0.3$) with label smoothing improves the performance on the test set further by 0.408% (from 76.56% to 76.968%) and reduces the generalization gap (from 2.54% to 1.5%). However, the impact of combining PGT with label smoothing can vary depending on the value of the hyperparameter ($\alpha$). **Row-7** With a PGT hyperparameter value of $\alpha = 0.05$, we notice the greatest performance improvement, 1.246% over the step scheduler test baseline and 0.656% over the cosine scheduler test baseline. **Row-8** Adding label smoothing to PGT ($\alpha = 0.05$) hurts performance even though it reduces the generalization gap.

![Figure 3: Norm vs iter. plots demonstrating layer characteristics (the zero out phenomena) and the efficacy of PGT over baseline. Each colour represents a different filter or feature vector of a particular layer of a non-BN variant of ResNet-18.](image1)

![Figure 4: Zeroing out of feature maps in the second layer non-normalized ResNet-18.](image2)

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Method</th>
<th>PGT ($\alpha$)</th>
<th>Train Acc. (%)</th>
<th>Train Diff (%)</th>
<th>Test Acc. (%)</th>
<th>Test Diff (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>Baseline</td>
<td>-</td>
<td>66.27</td>
<td>-</td>
<td>64.816</td>
<td>-</td>
</tr>
<tr>
<td>1024</td>
<td>PGT</td>
<td>0.92</td>
<td>66.62</td>
<td>+0.35</td>
<td>65.498</td>
<td>+0.682</td>
</tr>
<tr>
<td>512</td>
<td>Baseline</td>
<td>-</td>
<td>68.02</td>
<td>-</td>
<td>66.552</td>
<td>-</td>
</tr>
<tr>
<td>512</td>
<td>PGT</td>
<td>0.25</td>
<td>69.5</td>
<td>+1.48</td>
<td>67.236</td>
<td>+0.684</td>
</tr>
<tr>
<td>256</td>
<td>Baseline</td>
<td>-</td>
<td>68.86</td>
<td>-</td>
<td>66.796</td>
<td>-</td>
</tr>
<tr>
<td>256</td>
<td>GC</td>
<td>-</td>
<td>69.04</td>
<td>+0.18</td>
<td>67.064</td>
<td>+0.268</td>
</tr>
<tr>
<td>256</td>
<td>AGC</td>
<td>-</td>
<td>69.06</td>
<td>+0.2</td>
<td>67.298</td>
<td>+0.502</td>
</tr>
<tr>
<td>256</td>
<td>PGT</td>
<td>0.25</td>
<td>69.97</td>
<td>+1.11</td>
<td>67.814</td>
<td>+1.018</td>
</tr>
<tr>
<td>256</td>
<td>Baseline</td>
<td>-</td>
<td>68.86</td>
<td>-</td>
<td>66.796</td>
<td>-</td>
</tr>
<tr>
<td>256</td>
<td>GC+PGT</td>
<td>0.25</td>
<td>68.67</td>
<td>-0.19</td>
<td>67.088</td>
<td>+0.292</td>
</tr>
<tr>
<td>256</td>
<td>AGC+PGT</td>
<td>0.25</td>
<td>70.92</td>
<td>+2.06</td>
<td>68.856</td>
<td>+2.06</td>
</tr>
</tbody>
</table>

**Table 2:** Results for non-normalized ResNet-18 on ImageNet-1K. Best training and test accuracies are highlighted in red and blue respectively. Top differences in training and test accuracies are marked in yellow.
Table 3: Ablation study for ResNet-50 on ImageNet-1K.

<table>
<thead>
<tr>
<th>#Row</th>
<th>Scheduler</th>
<th>Label Smoothing</th>
<th>PGT (α)</th>
<th>Train Acc. (%)</th>
<th>Test Acc. (%)</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Step</td>
<td>✓</td>
<td></td>
<td>78.99</td>
<td>75.97</td>
<td>3.02</td>
</tr>
<tr>
<td>2.</td>
<td>Step</td>
<td>✓</td>
<td>0.3</td>
<td>79.56</td>
<td>76.49</td>
<td>3.066</td>
</tr>
<tr>
<td>3.</td>
<td>Cosine</td>
<td>✓</td>
<td></td>
<td>79.18</td>
<td>76.56</td>
<td>2.62</td>
</tr>
<tr>
<td>4.</td>
<td>Cosine</td>
<td>✓</td>
<td>0.1</td>
<td>78.81</td>
<td>76.98</td>
<td>2.112</td>
</tr>
<tr>
<td>5.</td>
<td>Cosine</td>
<td>✓</td>
<td>0.3</td>
<td>79.43</td>
<td>76.88</td>
<td>2.544</td>
</tr>
<tr>
<td>6.</td>
<td>Cosine</td>
<td>✓</td>
<td>0.1</td>
<td>78.47</td>
<td>76.96</td>
<td>1.502</td>
</tr>
<tr>
<td>7.</td>
<td>Cosine</td>
<td>✓</td>
<td>0.05</td>
<td>79.68</td>
<td>77.21</td>
<td>2.464</td>
</tr>
<tr>
<td>8.</td>
<td>Cosine</td>
<td>✓</td>
<td>0.1</td>
<td>77.69</td>
<td>76.39</td>
<td>1.3</td>
</tr>
</tbody>
</table>

6. Conclusion

PowerGrad Transform enables a significantly better fit to the dataset as measured by training and test accuracy metrics. With PGT, gradient behavior is enhanced and weights attain better values in normalized networks and degenerate states are avoided in non-BN networks. We provide theoretical analyses of the transformation. With different network topologies and datasets, we are able to show the potential of PGT and explore its impacts from an empirical standpoint. PGT helps the network to improve its learning capabilities by locating a more optimum convergence point and simultaneously speeds up training.

References

BOP: A Bitset-based Optimization Paradigm for Content-based Event Matching Algorithms

Wei Liang, Wanghua Shi, Zhengyu Liao, Shiyou Qian, Zhonglong Zheng, Jian Cao and Guangtao Xue

Zhejiang Normal University, Zhejiang, China.
Shanghai Jiao Tong University, Shanghai, China.

Abstract—Content-based publish/subscribe systems are widely used in many fields. Event matching is the core component to achieve fine-grained content-based data distribution. Many efficient algorithms have been proposed to improve event matching performance. However, in large-scale content-based publish/subscribe systems, event matching is still the performance bottleneck of the entire system due to the need to perform a lot of operations, such as additions, comparisons and bitmarkings. In this paper, we explore to convert various non-logical operations into efficient logical ones, and propose a bitset-based optimization paradigm (BOP) for matching algorithms. On the one hand, BOP can eliminate expensive operations in the matching process, greatly improving matching performance. On the other hand, BOP can stabilize the performance of matching algorithms, ensuring the quality of service of data distribution. We apply BOP to optimize two existing matching algorithms, namely TAMA and REIN. The experimental results show that BOP shortens the matching time of TAMA and REIN by more than 60%. In addition, the performance of optimized versions is more stable than the original matching algorithms.

Index Terms—Publish/subscribe, event matching, optimization, bitset

I. INTRODUCTION

The publish/subscribe system has been widely used in many fields, such as online advertising [1], mobile message push [2], information filtering [4], and content-based routing systems [5]. It provides a loosely coupled messaging architecture [6]. The publisher publishes messages (also called events) to the broker, and the subscriber submits his/her interest (often called subscriptions) to the broker. The broker matches each event with subscriptions and forwards the event to all interested subscribers. According to different subscription models, the publish/subscribe system can be roughly divided into two categories: topic-based and content-based. The first category is relatively simple, but the granularity is coarse and the expression ability is limited. The content-based publish/subscribe (CPS) system overcomes these shortcomings, and makes the system more flexible.

Event matching algorithm is the core component of CPS systems. Matching efficiency is critical, which directly affects the performance of the entire system. The design of high-performance event matching algorithm should consider time efficiency, subscription maintenance cost and memory consumption. For these three aspects, time efficiency is the major factor that affects the performance of event matching algorithm. Therefore, how to improve the efficiency of matching algorithm is a key problem.

With the continuous research and innovation of scholars, many efficient event matching algorithms have been proposed, such as PS-Tree [7], H-Tree [8], MO-Tree [9], TAMA [10], REIN [11], Siena [12], SCSL [13], HEM [14], PhSIH [15] and Comat [16]. These algorithms utilize different data structures, such as trees, tables and bloom filters, to index subscriptions to achieve high event matching speed. Different algorithms have their advantages and disadvantages.

According to the initial search target in the matching process, existing algorithms can be classified into forward matching algorithm such as TAMA [10] and OpIndex [17], and backward matching algorithm such as REIN [11], Ada-REIN [18] and GEM [19]. The forward matching algorithm directly searches matching subscriptions. They need to perform a lot of additions for counting-based algorithms and comparisons for tree-based algorithms. For example, TAMA [10] is built on an index table which is designed to locate all predicates that are satisfied by the given event value. For subscriptions containing each satisfied predicate, TAMA needs to perform a countering operation. On the other hand, the backward matching algorithm initially aims to search unmatching subscriptions. For example, REIN [11] first locates two lists of cells containing unsatisfied predicates for each event value and then traverses the cells to mark unmatching subscriptions in a bitset. Overall, these algorithms perform a large number of repetitive operations, which decreases the matching efficiency.

To improve the efficiency of existing matching algorithms, we propose a bitset-based optimization paradigm (BOP). The basic idea of BOP is manifested in two aspects. First, with bitsets, BOP can transform the expensive arithmetic operations in existing matching algorithms to efficient logical operations, which can make full use of the characteristics of hardware. Second, the status (matching or unmatching) of subscriptions can be marked in advance in bitsets, which avoids performing a large number of repetitive operations in the matching process. We formalize the optimization paradigm for forwarding matching algorithms. In addition, we extend it to support backward matching algorithm. Therefore, BOP is a general optimization method to improve the efficiency of most matching algorithms.

We apply BOP to optimize two existing matching algorithms, a forward matching algorithm TAMA [10] and a backward matching algorithm REIN [11], resulting in REIN-O and
TAMA-O respectively. Based on the original data structures of TAMA and REIN, BOP introduces bitsets to speed up event matching. We conducted extensive experiments to evaluate the effectiveness of BOP with different parameter settings, including subscription size, event size, predicate width and predicate attribute distribution. The experimental results show that REIN-O and TAMA-O achieve significant performance improvement. Compared with TAMA and REIN, REIN-O and TAMA-O reduce matching time by more than 60%. In addition, the performance of REIN-O and TAMA-O is more stable than that of TAMA and REIN.

The main contributions of this paper are as follows:

- We propose a bitset-based optimization paradigm called BOP, which aims to improve the efficiency of existing matching algorithms.
- We formalize the optimization paradigm and apply it to optimize two existing matching algorithms.
- We conduct a series of experiments to verify the optimization effect of BOP.

The remainder of this paper is organized as follows. Section II presents the background knowledge. Section III briefly discusses the related work. Section IV describes the design details of BOP. Section V analyzes the experimental results. Section VI concludes the paper.

II. PRELIMINARIES

In this section, we provide the basic knowledge of the event matching problem.

Attribute: An attribute $a_i$ represents the name of a data item, which appears in events and subscriptions. Each attribute in events has a value, expressed in the form of $< a_i, v >$ or $a_i = v$. The number of all attributes in the content space is called the dimensionality, denoted by $d$.

Event: Event is also known as message, consisting of multiple attribute-value pairs. Event is expressed as $E = \{a_1 = v_1, a_2 = v_2, \ldots, a_m = v_m\}$, where $m$ represents the number of attributes in the event. $m$ is called the event size in the paper. Note that each attribute appears only once in an event.

Predicate: A predicate defines a constraint on an attribute. An interval predicate is expressed as $P < a_i, lower, upper >$, where $a_i$ is an attribute, and lower and upper represent the left and right boundaries of the predicate, respectively.

Subscription: A subscription is composed of multiple interval predicates in the conjunctive normal form. A subscription $S$ is expressed as $S = \{P_1 \land P_2 \land \ldots \land P_k\}$, where $k$ represents the number of predicates in the subscription. $k$ is called the subscription size in the paper.

Match: Given an attribute-value pair $a_i = v_i$ and a predicate $< a_j, lower, upper >$, if $a_i = a_j \land v_i \in [lower, upper]$, the attribute value satisfies the predicate. Given an event $E$ and a subscription $S$, if each predicate in $S$ is satisfied by the corresponding attribute value of $E$, $S$ is a match of $E$.

Event Matching Problem: Given a set of $n$ subscriptions and an event $E$, the event matching problem is to search all the matches of $E$ from the subscription set.

III. RELATED WORK

In this section, we review the related work and classify existing algorithms into two categories: forward matching and backward matching. Then, for each category, we briefly introduce the matching process and discuss a representative algorithm. Finally, we describe the difference between BOP and existing work.

A. Algorithm Classification

In order to achieve high matching performance, an effective data structure for indexing subscriptions is necessary. The classic data structures include match tree [20], match table [21], binary decision graph [22] and Bloom filter [23]. According to the initial search targets, the matching algorithms based on these data structures can be divided into two categories: forward matching algorithm and backward matching algorithms.

B. Forward Matching Algorithms

The forward algorithm aims at matching subscriptions during the matching process, which can be further divided into two sub-categories: counting-based algorithms such as TAMA [10] and OpIndex [17], and tree-based algorithms such as PS-Tree [7] and H-Tree [8].

The counting-based matching algorithm sets a counter for each subscription. In the matching process, for each predicate $P$ satisfied by the event value, the counter of each subscription containing $P$ will be incremented by one. For a subscription, if the number of satisfied predicates recorded in the counter is equal to the subscription size, the subscription is a match of the event. The basic idea of the tree-based matching algorithm is to use the pruning ability of trees, where subscriptions are usually stored in leaf nodes. In the matching process, internal nodes are used to filter unmatching subscriptions, finally generating a set of candidate subscriptions. These subscriptions are accurately evaluated to obtain matching results. To better understand the forward matching algorithm, we discuss a representative algorithm TAMA.

TAMA [10] is a forward and counting-based matching algorithm. Its overall structure can be divided into two layers. The first layer is indexed on attributes. All attributes are organized into a linear table. The second layer adopts the hierarchical discretization method, which divides each predicate into multiple levels. Specifically, the value space of each attribute is divided into cells level by level. Different levels have different granularity. The upper cell is evenly divided into two smaller cells in the lower level. Thus, at the $i$-th level, the attribute space is divided into $2^i$ cells, and each cell at the $i$-th level will be separated into two cells at the $(i+1)$-th level. Given an interval predicate $P$, from the level that $P$ can contain the cell, the ID of the subscription containing $P$ is stored in at most two cells that are covered by $P$ at each level. Given an event value $v$, the cell containing $v$ stores the subscriptions whose predicates are satisfied by $v$. The counter of these subscriptions will be incremented by one.
C. Backward Matching Algorithms

The backward matching algorithm initially targets at unmatched subscriptions during the matching process, such as REIN [11] and GEM [19]. Given the set of subscriptions, when the unmatching ones are determined, the matches can be easily obtained. Generally, the working principle of backward matching algorithms is to search all unsatisfied predicates. For each unsatisfied predicate \( P \), all subscriptions containing \( P \) are marked as unmatching in a bitset. REIN is a representative backward matching algorithm, which is discussed in the following.

REIN transforms the matching problem into a rectangular intersection problem, and proposes an index structure to effectively solve the problem. REIN divides the value space of attributes into multiple cells and realizes the one-to-one mapping between predicate values and cells. When matching an event value, the list of cells containing unsatisfied predicates can be easily determined. These cells are iteratively processed by marking the unmatched subscriptions in a bitset. After processing all attributes, the unmarked bits represent matching subscriptions. Therefore, REIN mainly performs cell traversal and bit marking operations.

Existing algorithms perform a large number of repetitive operations, such as additions in TAMA. Compared with logical operations, these operations have higher costs from the perspective of hardware. To improve the performance of existing matching algorithms, we propose a general bitset-based optimization paradigm (BOP), which is applicable to most algorithms. BOP explores to convert costly non-logical operations into efficient logical operations by trading off space for time.

IV. DESIGN OF BOP

In this section, starting from the basic idea of BOP, we use bitset to optimize the forward algorithm first, and then extend it to optimize the backward algorithm.

A. Basic Idea

When the number of subscriptions is large, for most algorithms, matching an event needs to perform a large number of repeated operations, such as comparisons, additions or bitmarkings. As discussed in Section III, the forward counting-based matching algorithm calculates the number of satisfied predicates for each subscription. When there is only one unsatisfied predicate in a subscription of size \( k \), it is redundant and inefficient to count the \( k - 1 \) satisfied predicates. Therefore, when the predicate width is wide and the subscription size is large, the performance of the counting-base matching algorithm will decline. The backward matching algorithm searches unsatisfied predicates and marks all subscriptions containing them. When there are multiple unsatisfied predicates in a subscription, the subscription will be marked repeatedly. Except for the first time, the subsequent marks are redundant and inefficient. Therefore, the backward matching algorithm performs poorly in high-dimensional content space and small-width predicates.

As a result, from the perspective of hardware, the basic idea of BOP is to convert a large number of expensive operations performed by the matching algorithm into more efficient logical operations. For this purpose, we introduce bitset as a caching mechanism to reduce some repeated operations. In addition, based on bitsets, BOP performs as many logical operations as possible to improve matching performance.

When designing BOP, two guidelines should be followed. First, BOP should be general, which means that BOP should be able to optimize backward and forward matching algorithms. Second, BOP should be non-intrusive, which means that BOP should not change the underlying data structure and matching process of existing algorithms.

B. BOP for Forward Counting-based Algorithm

Most forward counting-based algorithms index predicates defined on an attribute separately. This type of algorithms can only retrieve the satisfied or unsatisfied predicates on single attribute. When matching, the attributes in the event are processed one by one. For each attribute, when finding a satisfied predicate, all subscriptions containing the predicate can be marked in a bitset, instead of incrementing the counters. In this way, counting operations can be replaced with marking operations.

In addition, in the content space with \( d \) attributes, subscription size \( k \) is generally far smaller than \( d \). In other words, subscriptions usually define predicates on some attributes, with most attributes in the content space having null predicates. From the semantics of event matching, the event value matches all null predicates. Therefore, for each attribute \( a_i \ (i \in [1, 2, \ldots, d]) \), BOP sets a bitset \( B_{null}^i \) to record the subscriptions that do not define predicate on \( a_i \).

The matching process with BOP can be formalized as follows:

\[
B_{result} = \bigcap_{i=1}^{d} (B_{null}^i \cup B_{match}^i)
\]  

(1)

where \( B_{result} \) represents the result bitset, \( d \) is the dimensionality of content space, \( B_{null}^i \) is the bitset used to mark subscriptions that have no predicate defined on attribute \( a_i \), and \( B_{match}^i \) is the bitset that marks all subscriptions whose predicates defined on attribute \( a_i \) are satisfied.

C. Extension for Backward Marking-based Algorithm

BOP can also be extended to optimize the backward marking-based algorithm. Backward matching aims at finding all unsatisfied predicates and marking the corresponding subscriptions as unmatching in a bitset. Following the optimization paradigm defined in Formula (1), backward matching should be first transformed into forward matching, and then logical AND operations are performed. Specifically, the bitset associated with each attribute is initially set to all 1. In the matching process, all unmatching subscriptions found are marked as 0 in the bitset for each attribute. Then, logical AND operations are performed based on the bitset of all attributes. The bits with value 1 in the result bitset represent the matching subscriptions.
D. Implementation Considerations

To apply BOP, there are two implementation considerations. The first is to reduce the memory consumed by bitsets in the matching process, and the second is to efficiently process logical operations.

Intuitively, BOP needs to set $d$ bitsets in the event matching process, and obtains the result bitset by performing logical operations on the $d$ bitsets. This will consume a lot of memory in high-dimensional scenarios. By analyzing the principle of BOP, we find that only two bitsets are needed to iterate in the implementation. Actually, processing an attribute consists of two steps: marking all subscriptions containing satisfied predicates and performing a bitset AND operation. Therefore, it is not necessary to maintain one bitset for each attribute at the same time, but only two bitsets in the matching process. One bitset is used to store the result after processing all previous attributes, and the other bitset stores the subscription matching status on the current attribute.

An event of size $m$ has no value on $d - m$ attributes, which are called null attributes. For each null attribute $a_i$, the corresponding $B_i^{null}$ is all 0, thus no logical OR operation is required. Thus, there is no need to perform $d - m$ OR operations. BOP first performs a logical OR operation between $B_i^{null}$ and $B_i^{match}$ for each attribute in the event, so there will be $m$ bitset OR operations. Then, BOP performs $d$ bitset AND operations. A total of $d + m$ logical operations are required. However, in the implementation, we can eliminate bitset OR operations. For the $m$ attributes contained in the event, we can copy $B_i^{null}$ first, and mark the found matching subscriptions directly in $B_i^{null}$, without setting the bitset $B_i^{match}$. In this way, BOP only needs to perform $d$ bitset AND operations.

E. Complexity Analysis

BOP needs to set a bitset for each attribute, which has the space complexity of $O(nd)$, where $n$ is the number of subscriptions and $d$ is the dimensionality of the content space. In addition, two bitsets are required in the matching process. For subscriptions of size $k$, the insertion time is increased by $O(d - k)$ due to the marking operations on the null attributes without defining predicates. In the matching process, counting operations are replaced with marking operations, so the number of operations does not change. BOP needs to perform the additional $d$ bitset AND operations.

V. EXPERIMENTS

To demonstrate the effectiveness of BOP for event matching algorithms, we select two different matching algorithms to optimize, namely REIN [11] and TAMA [10].

A. Test Benches

1) BOP on TAMA: According to the optimization paradigm defined in Formula (1), BOP can be directly applied to TAMA, which results in a new variant called TAMA-O. When inserting a subscription $S$, it is necessary for TAMA-O to mark $S$ as matching in the bitset $B_{null}$ of each null attribute on which $S$ does not define a predicate. When matching an event $E$, TAMA-O processes the attributes in $E$ one by one. First, a copy of $B_{null}$ is made, denoted by $B_{null}^′$, for each attribute. Then, based on the original data structure of TAMA, TAMA-O searches satisfied predicates for each event attribute and marks all subscriptions containing these predicates as matching in $B_{null}^′$. Finally, a logical AND operation is performed between the result bitset $B_{result}$ and $B_{null}^′$. After processing all attributes in $E$ and all null attributes, the unmarked bits in $B_{result}$ represent the matching subscriptions of $E$.

2) BOP on REIN: Since REIN is a backward marking-based matching algorithm, we first consider transforming it into a forward matching, generating a variant called REIN-F. Specifically, REIN-F marks (set to 0) all subscriptions containing unsatisfied predicates in the bitset initialized to all 1, and then obtains subscriptions that match on all attributes through bitset AND operations. In the implementation, REIN-F does not need to actually perform AND operations. For all attributes, the same effect can be achieved by marking 0 on the same bitset, similar to REIN. Based on the data structure of REIN, BOP further optimizes REIN-F by searching satisfied predicates on each attribute. In this way, REIN-F works in the same way as the counting-based matching algorithm. Therefore, REIN-F can be optimized by BOP, resulting in a new variant called REIN-O.

B. Setup

For simplicity, we consider the value space of each attribute as [0,1]. Event values and predicate values are generated based on uniform distribution. Attributes are selected based on Zipf distribution with different setting of parameter $\alpha$. Since the performance of matching algorithms is affected by many parameters, we evaluate the effects of subscription size $k$, event size $m$, predicate width $w$, predicate attribute distribution $\alpha$, and the number of subscriptions $n$. The settings of these parameters are shown in Table I. By default, we set $n$ to 1 million, $d$ to 50, $w$ to 0.4, $k$ to 6, and $m$ to 20. In each experiment, 1,000 events are matched and the average matching time is calculated. All the experiments were conducted on a server with 128 1.4GHz CPUs, which runs Ubuntu 18.04.6 and Linux kernel 5.4.0-136. All code is written in C++ language and compiled by g++ with version 7.5.0.

C. Results and Analysis

1) Effect of Subscription Size $k$: We set $k$ to 4, 6, 8 and 10 in the experiment. The experimental results of the four tested algorithms are shown in Figure 1. We can see that BOP significantly improves the performance of TAMA and
REIN. Compared with TAMA, TAMA-O reduces the matching time by 61% on average. In addition, as \( k \) increases, TAMA-O performs more stable. Similarly, REIN-O is more efficient and stable than REIN, reducing the matching time by 71% on average. The reason why BOP achieves better performance is that it converts expensive non-logical operations into logical operations, which is more efficient for hardware.

2) Effect of Event Size \( m \): In this experiment, we set \( m \) to 10, 20, 30 and 40. The experimental results are shown in Figure 2. Overall, the matching time of REIN decreases with the increase of \( m \). When fixing the subscription size, events of larger size have more matching subscriptions, which is conducive to improving the performance of REIN. On the contrary, the average matching time of TAMA increases with \( m \). With more matching subscriptions, TAMA needs to perform more counting operations. Compared with TAMA, TAMA-O reduces the matching time by 62% on average. Similarly, compared with REIN, REIN-O reduces the average matching time by 61%.

3) Effect of Predicate Width \( w \): In this experiment, we set \( w \) from 0.1 to 0.9. The experimental results are shown in Figure 3. Overall, the matching time of TAMA increases with the increase of \( w \), while REIN is the opposite. On average, the matching time of REIN-O is reduced by 66% compared with REIN. When \( w=0.1 \), the matching time is reduced by up to 89%. Similarly, the matching time of TAMA-O is much smaller than that of TAMA. When \( w=0.1 \), compared with TAMA, the matching time of TAMA-O is reduced by 79%. With the increase of the matching probability, there are more and more 0s set in the bitset. This is not conducive to the logical OR operations. Thus, the optimization effect of BOP continues to decline with the increase of \( w \), resulting in an average reduction of 60% in the matching time.

4) Effect of Attribute Distribution \( \alpha \): In the experiment, attributes follow the Zipf distribution, where the parameter \( \alpha \) is set to 0, 0.25, 0.5, 0.75 and 1. The experimental results are shown in Figure 4. Overall, the matching time of the four algorithms is basically unchanged with the increase of \( \alpha \). This is because the matching probability of subscriptions does not change. BOP can optimize TAMA, reducing the matching time by 60% on average. Similarly, REIN-O reduces the matching time by 65% compared with REIN.

5) Effect of Subscription Number \( n \): In this experiment, we set \( n \) from 1M to 9M. The experimental results are shown in Figure 5. The matching time of the four algorithms increases with \( n \) because the total workload increases. TAMA-O and REIN-O perform better than their baselines, reducing the matching time by 69% and 66% respectively on average.
6) Performance Stability: In this experiment, we evaluate the performance stability of the four algorithms in terms of the 95th percentile (P95) of matching time. With the default parameter settings, we change $k$, $m$, $w$ and $\alpha$ respectively. The experimental results are shown in Figure 6. With BOP, TAMA-O and REIN-O perform more stable than their baselines. For example, in the first group of experiments, the P95 value of REIN, REIN-O, TAMA and TAMA-O is 12.27, 4.43, 4.90 and 1.87 ms respectively. Similar conclusions can be reached in other groups of experiments. Therefore, we verify that BOP is beneficial to improve and stabilize matching performance.

D. Memory Usage

In this experiment, we set each parameter as the default value to measure the memory consumption of the four algorithms. The experimental results are shown in Figure 7. Compared with TAMA, TAMA-O does not increase memory consumption. REIN-O increases memory usage by about 39% on the basis of REIN. This is mainly because a bitset is configured for each attribute. Considering that REIN-O reduces the matching time by more than 60%, BOP well achieves the goal of exchanging space for time.

VI. CONCLUSION

By summarizing the working principles of existing matching algorithms, we propose BOP to improve their performance. The basic idea of BOP is to convert a large number of non-logical operations into logical ones. The experimental results show that BOP can improve and stabilize the performance of matching algorithms. In the future, the optimization effect of BOP can be further improved by dividing attributes into groups and setting a bitset for each group.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (61772334).

REFERENCES

Transient Data Caching Based on Maximum Entropy Actor-Critic in Internet-of-Things Networks

Yu Zhang
School of Computer and Electronic Information
Guangxi University
Nanning, China
zhany.03@qq.com

Ningjiang Chen*
School of Computer and Electronic Information
Guangxi University
Nanning, China
chnj@gxu.edu.cn

Siyu Yu
School of Computer and Electronic Information
Guangxi University
Nanning, China
gaiusyu6@gmail.com

Abstract—With the rapid development of the Internet of Things (IoT), a massive amount of transient data is transmitted in edge networks. Transient data is highly time-sensitive, such as monitoring data generated by industrial devices. Due to their inefficiency, traditional caching strategies in edge networks are inadequate for handling transient data. Thus, to improve the efficiency of transient data caching, we construct a freshness model of transient data and propose a maximum entropy Actor-Critic based caching strategy, TD-MEAC—which can improve the freshness of cached data and reduce the long-term caching cost. Simulation results show that the proposed TD-MEAC achieves a higher cache hit rate and maintains a higher average freshness of cached transient data compared with the existing DRL and baseline caching strategies.

Keywords—Internet of Things; edge networks; transient data caching; maximum entropy Actor-Critic

I. INTRODUCTION

The Internet of Things (IoT) development has caused exponential device growth and generated vast data at the edge network. The data generated by IoT devices are used in extensive applications across various domains. Most IoT data is transient, with short-lived, time-sensitive, and dynamic characteristics. For example, some health monitoring sensors produce specific health parameters only valid for a few minutes. The invalid data can affect the reliability and accuracy of application decisions.

Caching transient data at the edge network can accelerate response times and reduce network traffic, thereby improving Quality of Service (QoS) and Quality of Experience (QoE)[1]. For instance, in a vehicle traffic monitoring system, collecting and timely updating the status data of the monitoring objects is crucial. Caching this valid status data at the edge nodes eliminates the need for IoT devices to respond to data requests, thereby reducing the load on the return path. Additionally, for some latency-sensitive applications, such as intelligent power distribution monitoring systems, responding to requests by edge nodes can more quickly isolate device failures[2].

Most edge caching studies only apply to data that remains valid in the long term. Researchers [3]-[6] propose corresponding caching schemes for different caching scenarios suitable for long-term valid data. However, due to IoT data's transient nature, the data gradually becomes obsolete and invalid.

The main challenges transient data poses to existing caching strategies are summarized as follows:

- Data freshness. The caching of transient data needs to consider data freshness, which depends on the application's timeliness and lifecycle requirements. Data becomes stale with time, resulting in freshness loss. Thus, caching transient data requires balancing caching benefits and freshness loss.

- Frequent replacements. Compared to long-lived data, the caching of transient data requires more frequent cache replacements, which may increase the energy consumption of edge nodes.

Therefore, caching strategies for long-lived data are not suitable for transient data.

Some studies have sought to improve traditional caching strategies to adapt to the transient data. A cache replacement strategy based on data freshness is proposed in [7], which selects data with the shortest remaining lifespan for replacement when cache replacement occurs. Reference [8] suggested the expected validity of data lifecycles, which predicts the number of requests received by cached data in the remaining lifespan and evicts data with the lowest expected number of requests for cache replacement. References [9][10] investigate transient data caching in vehicular networking and transient data caching on routers, respectively. And [11] used the least recently used replacement strategy and considered the transient nature of the data to determine whether to cache it by setting a freshness threshold.

Although these methods consider data transience, they require prior knowledge, such as popularity and network topology. However, edge networks are dynamic, with the position, connection status, resource distribution, application scenarios, and demands of IoT devices constantly changing, making it challenging to provide prior knowledge.

Deep reinforcement learning (DRL)[12] has recently attracted widespread attention as a new machine learning paradigm. DRL uses deep neural networks to represent and process high-dimensional state space, improving decision-making capabilities in large-scale state space. Without prior knowledge and state features of edge networks, DRL can generate a series of mappings between network states and caching actions through multi-round interactions with edge networks, making DRL an excellent method for solving the caching problem of transient data in edge networks.

In [13], DRL was first used to solve the caching problem of transient data in edge networks. An Actor-Critic method is trained in an asynchronous and parallel manner, allowing caching decisions to be made without a priori knowledge.
Researchers [14] proposed a hierarchical network architecture caching system, formulating the problem as a Markov decision process (MDP) and designing a policy optimization solver to obtain the optimal policy. Reference [15] considered the limited caching capacity and used a distributed proximal policy optimization (DPPO) algorithm to optimize the allocation of cached data and improve training speed.

Unlike the above methods, this paper proposes a transient data cache strategy based on the Maximum Entropy Actor-Critic, TD-MEAC, which is based on the traditional Actor-Critic algorithm and considers the randomness of cache actions. It improves the exploration ability and can better adapt to the dynamic edge environment. The main contributions of this paper are as follows:

- A caching strategy for transient data is proposed, considering the data lifecycle and propagation delay to construct a freshness model for the data. A cost function that balances the freshness loss of the data and the cost of acquiring data items is also presented, which can reduce the long-term acquisition cost of transient data.
- Considering the practical limitations of the caching scenario, the caching replacement problem for transient data is formulated as MDP. The freshness of data items is viewed as a part of the state space, and the efficiency function is applied to the reward function. The Actor-Critic algorithm with maximum entropy interacts with the environment and finds the optimal caching decision.

This paper compares the TD-MEAC with other approaches, including classical and DRL-based caching strategies. Simulation results show that the proposed TD-MEAC achieves a higher cache hit rate and maintains a higher average freshness of cached transient data.

II. SYSTEM MODEL

A. Network Architecture

This paper considers a centralized caching scenario for a group of IoT devices covered by edge nodes in edge networks, as shown in Fig. 1. This scenario consists of IoT devices, applications, and edge nodes.

Edge nodes randomly distribute N IoT devices within their coverage area, with each IoT device capable of generating only one data item type. The corresponding edge node can cache these data items. When an IoT application sends a data request, the edge node checks its cache unit to see if there is a related data item. If there is, and the freshness of the data item meets the requirements, the edge node responds to the request. Otherwise, the edge node forwards the request to the corresponding IoT device, which generates a new data item and answers via the return link.

B. Freshness model of data items

Freshness reflects the staleness of transient data, and a lower freshness indicates a more stale data item. Worn data items can reduce the accuracy and timeliness of applications. Different applications have additional requirements for data freshness. Factors that affect data freshness include data collection and processing time, transmission delay, and timeliness of data processing.

![Fig. 1 The network architecture of edge caching](image1)

As shown in Fig. 2, the data item is still valid and can be used to respond to the request. In contrast, in Fig. 3, the cached data item has expired, and the carried information is invalid, thus cannot be used as a response to the request.

C. Cost Function

The caching of transient data at edge nodes results in a lag, causing the cached data to have a non-zero data age compared to directly acquired data from IoT devices. Consequently, the cache results in a loss of data freshness. The cost of freshness loss for a newly generated data item from IoT devices can be considered zero, while the cost of freshness loss for a cached data item i is related to its data age. Therefore, the cost of freshness loss denoted as Costloss, is defined as follows:

\[
\text{Cost}_{\text{loss}} = \begin{cases} 
\frac{t'_{\text{life}} - t'_{\text{age}} - t'_{\text{del}}}{t'_{\text{life}}} , & t'_{\text{age}} > t'_{\text{life}} + t'_{\text{del}} \\
0 , & \text{otherwise}
\end{cases}
\]
The acquisition cost of data denoted as Cost\( g_1 \), depends on where the data is obtained. The cost of acquiring data from an edge node is denoted as Cost\( g_2 \), while the cost of acquiring data from IoT devices is denoted as Cost\( g_3 \). The generation of data items by IoT devices consumes device energy, and there is a transmission delay from IoT devices to edge nodes. Thus, it is generally true that Cost\( g_3 < \text{Cost}_{g_2} \).

When an edge node responds to a request, the cost consists of two parts: the cost of freshness loss Cost\( \text{loss} \), and the cost of acquiring data items by the edge node Cost\( g_1 \). When IoT devices respond to the request, there is only the cost of acquiring data items Cost\( g_2 \). Both costs affect the benefit of caching transient data. To balance the two costs, the cost function, denoted as Cost, is defined as follows:

\[
\text{Cost} = \psi \text{Cost}_{\text{loss}} + \varphi \text{Cost}_i\quad \psi + \varphi = 1
\]  

(3)

Where \( \psi \) and \( \varphi \) respectively denote the weighting coefficients for the cost of freshness loss and the acquisition cost of data items.

To evaluate the effectiveness of caching, we define the benefit function \( U \). Minimizing the cost is equivalent to maximizing the benefit since the cost of caching is inversely proportional to its benefit. To ensure that the benefit function is always non-negative, we define the constant \( a \text{Cost}_{g_2} + \beta \) as the baseline benefit value. Therefore, the benefit function is defined as follows:

\[
U = \psi(\text{Cost}_{g_2} - \text{Cost}_{\text{loss}}) + \varphi (1 - \text{Cost}_i)
\]  

(4)

III. PROBLEM FORMULATION

A. Cache Replacement Model

The state of edge nodes is determined solely by the previous cache state and the previous caching action. Therefore, the cache replacement process can be represented as MDP. Typical MDP consists of a quintuple \( \{S, A, P, R, \gamma\} \). In the transient data cache model, the number of requests processed by an edge node in a single time step is unknown. If the unknown number of requests is regarded as part of the MDP, the complexity of the model will increase exponentially. Therefore, this paper regards processing a single request by an edge node as a decision cycle and replaces the time step with a decision cycle. And the quintuple for the \( n \)-th decision cycle can be represented as \( \{s_n, a_n, p(s_{n+1} | s_n, a_n), r_n, \gamma\} \).

The state space \( S \) represents a finite set of states for the caching process of edge nodes. The state features are split into two parts: the data items already cached in the cache space and the requests from IoT applications. The cache space size is defined as \( C \), and \( C \) binary tuples can represent the data item features \( d_i = \{\text{CID}_i, \text{freshness}_i\} \). The request is represented by \( d_0 = \{\text{CID}, \text{requestTime}\} \). Thus, the state \( s_n \) for the \( n \)-th decision cycle can be represented as \( s_n = \{d_0, d_1, d_2 \ldots d_c\} \).

The action space \( A \) represents a finite set of caching actions for edge nodes. The action taken by the edge node in the \( n \)-th decision cycle is represented by \( a_n \). The size of the action space is proportional to the cache space and is related to the selected caching action. The larger the action space, the greater the computational complexity of the model. In resource-constrained edge nodes, to limit the size of the action space, it is stipulated that at most one cache replacement occurs in a decision cycle. The action space is represented as \( A = \{0, 1, 2, 3, \ldots, C\} \). Here, \( a_n = 0 \) represents skipping the cache replacement process while \( a_n = 1 \ldots C \) represents replacing the new data item with the \( a_n \)-th data item in the cache space.

The state transition probability \( P(s_{n+1} | s_n, a_n) \) represents the probability of the following decision cycle state \( s_{n+1} \), given the state-action combination \( \{s_n, a_n\} \).

The reward function \( R \) is an instantaneous utility function. \( r_n \) represents the expected instantaneous reward for the state-action combination in the \( n \)-th decision cycle. The utility function \( U \) serves as a suitable indicator for training DRL, balancing freshness loss cost and data acquisition cost. Define reward function \( R \) as:

\[
R(s_n, a_n) = E[r_n | s_n, a_n \sim P], \text{where } r_n = U
\]  

(5)

In addition to the immediate reward, the impact of future rewards on the current decision is also considered. A discount rate \( \gamma \) is used to reconcile the cumulative rewards of the reward sequence. The smaller the \( \gamma \), the more biased the immediate reward is. Define the cumulative rewards \( G_n \) starting from the \( n \)-th decision cycle as:

\[
G_n = \sum_{i=n}^{\infty} \gamma^i r_{n+i}
\]  

(6)

The goal of MDP is to find an optimal caching policy \( \pi^\star \) that maximizes the long-term reward under this policy, as shown below:

\[
\pi^\star = \arg\max E[G_n]
\]  

(7)

B. TD-MEAC: Transient Data Caching Strategy Based on Maximum Entropy Actor-Critic

To enhance the exploration capability of the strategy, the randomness of cached actions is considered when selecting them, and the maximum entropy policy is used to improve the solution to the problem. The improved optimal policy \( \pi^\star \) is defined as:

\[
\pi^\star = \arg\max E[G_n + \alpha \sum_{i=0}^{\infty} \gamma^i H\left(p\left(s_{i+1} | \pi(s_i)\right)\right)]
\]  

(8)

Here, \( p\left(s_{i+1} | s_i\right) \) represents the probability distribution of cached actions taken under the marginal state state \( s_i \). The function \( H(p\left(s_{i+1} | s_i\right)) \) represents the entropy of the cached actions, which measures the randomness of the caching strategy. Increasing entropy can give the marginal node a more robust exploration capability and thus better discover potential high-reward policies. \( \alpha \) is the temperature parameter, which indicates the relative importance of the entropy term to the cumulative reward.

To calculate the optimal policy \( \pi^\star \), this paper defines the action value function \( Q^\pi(s,a) \) and state value function \( V^\pi(s) \). \( Q^\pi(s,a) \) measures the expected cumulative reward obtained by the marginal node based on the caching policy \( \pi \),
considering the entropy of the cached actions, and can be expressed as:

\[
Q^s(s_n, a_n) = E\left[ r(s_n, a_n) + \gamma V^s(s_{n+1}) \right] s_{n+1} \sim p(s_n, a_n)
\] (9)

\[
V^s(s) \text{ measures the expected cumulative reward obtained by the marginal node based on the caching policy } \pi \text{ after executing the cached actions, considering the entropy of the cached actions, and can be expressed as:}
\]

\[
V^s(s_n) = \pi(s_n) [Q^s(s_n, a) - \alpha \log \pi(a|s_n) a \sim \pi] 
\] (10)

Two independent neural networks are used to approximate the value function \(Q^s(s, a)\) and the policy function \(\pi(\cdot | s)\), denoted as \(Q_{\omega_1}(s, a)\) and \(\pi_{\theta}(a|s)\), respectively, where \(\omega\) and \(\theta\) are the parameters of the two networks. The policy network \(\pi_{\theta}(a|s)\) outputs a C-dimensional vector, representing the probability distribution of each cached action. The value network \(Q_{\omega_1}(s, a)\) outputs a score for the cached action. The convergence objective is achieved through stochastic gradient descent during policy evaluation and improvement.

To mitigate the overestimation caused by bootstrapping during the neural network training process, this strategy employs two value networks, \(Q_{\omega_1}(s, a)\) and \(Q_{\omega_2}(s, a)\), and defines two target value networks, \(Q_{\omega_1}(s, a)\) and \(Q_{\omega_2}(s, a)\), with the same structure as the former but with different parameters. To compute the TD target \(\hat{y}_n\), the minimum value between the two target value networks is taken and denoted as:

\[
\hat{y}_n = r + \gamma \left( \min_{i=1,2} Q_{\omega_i}(s_n, a_n) - \alpha \log \pi_{\theta}(a_n|s_n) \right), \quad a_n \sim \pi_{\theta}(a_n|s_n) 
\] (11)

The TD error, defined as \(\delta^i\), can be represented as the difference between the value network and the TD target, as shown below:

\[
\delta^i_n = Q_{\omega_i}(s_n, a_n) - \hat{y}_n, \quad i = 1, 2 
\] (12)

The workflow of the TD-MEAC caching strategy is illustrated in Fig. 4 and described as follows:

- Given the current state \(s_n\), the edge node uses the policy network \(\pi_{\theta}(\cdot|s_n)\) to obtain a cached action \(a_n\). The agent executes the action, and the environment provides the reward \(r(s_n, a_n)\) and the new state \(s_{n+1}\). The transition \((s_n, a_n, r(s_n, a_n), s_{n+1})\) is then stored in the experience buffer \(D\).
- A batch of transitions \(T\) is randomly sampled from the experience buffer \(D\), and the TD error of each transition is computed.
- The value and policy network parameters are updated. When updating the parameters of the value network, the objective function is defined as follows:

\[
J(Q_{\omega_i}) = E\left[ \frac{1}{2} (Q_{\omega_i}(s_n, a_n) - r(s_n, a_n) + \gamma \bar{V}^s(s_{n+1}))^2 \right] 
\] (13)

Here, \(\bar{V}(s_{n+1})\) represents the expected action distribution obtained by sampling experiences from the experience buffer using the target value network and (9). When updating the parameters of the policy network, the objective function is defined as follows:

\[
J(\pi_{\theta}) = E\left[ \alpha \log(\pi_{\theta}(a|s_n)) - \min_{i=1,2} Q_{\omega_i}(s_n, a) \right| s_n \sim T, a \sim \pi_{\theta}] 
\] (14)

The value network parameters \(\omega_1\) and \(\omega_2\) are updated using the stochastic gradient descent method based on (9) and (10). Then, the target value network parameters \(\bar{\omega}_1\) and \(\bar{\omega}_2\) are updated using the weight-based method.

Different temperature parameters must be used depending on the exploration stage. When the policy has essentially completed the exploration of a region and the optimal action has been determined, \(\alpha\) should be reduced. Conversely, when the procedure begins exploring a new area, \(\alpha\) should be increased to explore more space in search of the optimal action. Consequently, during policy updates, the temperature parameter \(\alpha\) is updated synchronously with the constraint formula proposed in [16] to adjust automatically, as shown in (15):

\[
J(\alpha) = E\left[ -\alpha \log(\pi(a|s_n)) + \bar{H} \right| a_n \sim \pi_{\theta}] 
\] (15)

Here, \(\bar{H}\) is a constant vector that represents the hyperparameter of the target entropy.
IV. EXPERIMENTS AND EVALUATION

A. Simulation Setting

The simulations were conducted on a workstation with an AMD Ryzen 7 3700 X 8-Core CPU and 32 GB RAM. The edge node’s cache space was set to 100 ($C = 100$), with 200 IoT devices within the coverage range. The data item’s lifespan was randomly sampled from 5 to 20 time steps, and the delay in network propagation was uniformly distributed between one to three timesteps. Fifty thousand requests were generated using the Zipf distribution with variable parameter $\epsilon$ ranging from 0.9 to 1.7.

Regarding the TD-MEAC neural network model, both the value and policy networks were set to two hidden layers. Each hidden layer had 64 neurons, and the relu function was the activation function between the layers. The learning rate was initialized at 0.0001, the discount rate $\gamma$ was set to 0.99, the initial value of the temperature parameter $\alpha$ was 0.2, and the weight coefficient $\psi$ of the cost was initialized to 0.6. Each batch contained 256 experiences during training, and the experience buffer size was set to 5000.

B. Results and Discussions

This paper compared TD-MEAC with three representative caching schemes, including a baseline caching strategy: LFF[7], and two DRL-based methods: DRL-Cache[13] and IoT-Cache[15].

This section analyzes the impact of different parameter indicators on cache strategies from multiple aspects.

1) Cache capacity: Fig. 5 and Fig. 6 illustrate the impact of varying cache capacity on the cache hit ratio and average freshness of cached items. It can be observed that the cache hit ratio increases, and the average freshness decreases with the rise of cache capacity. For transient data, cache misses are likely to occur if the capacity is small, resulting in a low cache hit ratio. And transient data experiences freshness loss once it is cached until invalid. Larger cache capacity results in fewer cache replacements, reducing the average freshness of cached items.

2) Request rate: As depicted in Fig. 7, the cache hit ratio of the cache strategy increases as the request rate rises from 2 to 10 requests/timestep. However, the cache hit ratio will
not continue to grow, and when the request rate exceeds ten requests/timestep, the cache hit ratio tends to stabilize. Fig. 8 shows the average freshness of cached data items increases as the request rate increases. Due to the increase in request rate, the number of cache replacements also increases, and invalidated cache data items are more likely to be replaced, thus increasing the average freshness. The average freshness stabilizes when the request rate exceeds ten requests/timestep. Compared with other cache strategies, the proposed TD-MEAC performs better at different request rates.

3) Weight factor \( \psi \): The weight coefficients in the cost function play an essential role in cache strategies. \( \psi \) represents the weight assigned to the cost of freshness loss. As shown in Fig. 9, the cache hit ratio based on the DRL-based cache strategy decreases as \( \psi \) increases from zero to one. This is because the cache strategy tends to retrieve new data from the IoT devices to avoid the cost of freshness loss associated with using cached data. As \( \psi \) increases, the IoT devices generate more data, decreasing the cache hit ratio. Fig. 10 presents a clear trend of the average freshness. As the proportion of fresh data retrieved from the IoT devices increases, the edge nodes replace cached data items with fresh ones of the same CID, resulting in an overall increase in the average freshness.

To investigate the inherent logic of different strategies in the transient data caching process, we recorded the response process of edge nodes to requests. We classified the requested data items according to their lifecycles. The cache hit ratio is shown in Fig. 11. Data items with longer lifecycles tend to have a higher cache hit ratio. For data items with lifecycles between 10 and 16, the cache hit ratio of the TD-MEAC strategy is superior to that of DRL-Cache and IoT-Cache. As for data items with other lifecycles, the performance of the three DRL-based strategies is comparable.

Finally, we recorded the long-term average cost as the number of requests changed. As shown in Fig. 12, the TD-MEAC strategy’s average cost converges to the lowest point at around 28,000 requests and maintains relatively stable performance afterward. TD-MEAC outperforms DRL-Cache and IoT-Cache strategies in reducing long-term average costs.

The performance of the proposed TD-MEAC strategy is still better than that of the DRL-Cache and IoT-Cache strategies. Part of the reason is that the TD-MEAC uses the method of entropy regularization and the optimal strategy \( \pi^* \) used contains the entropy of the cache actions. Entropy regularization can encourage the model to balance the predicted distribution of outputs and improve the convergence speed of the model. In addition, the TD-MEAC uses two independent value networks to reduce overestimation problems, making the learning process more stable.

V. CONCLUSION

This paper addressed the issue of caching transient IoT data in edge networks. A freshness model was established based on the transient data’s lifespan and latency, and a cost function was proposed that integrated data freshness and retrieval Cost. To make the approach suitable for dynamic edge environments, we developed a maximum entropy Actor-Critic-based caching strategy, TD-MEAC, reducing reliance on prior knowledge. The performance of TD-MEAC was evaluated experimentally and compared to other strategies. Experimental results indicate that TD-MEAC is better than other caching strategies regarding cache hit ratio, data freshness, and reducing long-term cache costs.

VI. ACKNOWLEDGMENTS

The work described in this paper is supported by the National Natural Science Foundation of China (No. 62162003) and the Nanning Science and Technology project (No. 20221031).

REFERENCES


Branchy-TEE: Deep Learning Security Inference Acceleration Using Trusted Execution Environment

Yulong Wang, Kai Deng, Fanzhi Meng, Zhi Chen, Mingyong Yin*
Institute of Computer Application
China Academy of Engineering Physics
Mianyang, China
{wangyulong, dengkai, mengfz, chenzen, yinmy}@caep.cn

Run Yang
School of Cyberspace Science
Harbin Institute of Technology
Harbin, China
runyoung@hit.edu.cn

Abstract—Deep Learning as a Service (DLaaS) has become a remarkable trend in modern data-driven online services. Both data holders and service providers need to build on trust in third-party cloud infrastructure platforms. However, once the trust is broken, data holders’ sensitive data and service providers’ intellectual property rights will face significant security and privacy risks. In this paper, we propose a secure and efficient inference framework for deep learning in untrustworthy cloud platforms, termed Branchy-TEE, which aims to protect the confidentiality and integrity of data and models of multiple participating actors throughout the inference process using the Trusted Execution Environment (TEE). Branchy-TEE dynamically loads the inference network into the TEE on demand based on early-exit mechanism, expecting to break the hardware performance bottleneck of the TEE. Moreover, a joint training method based on knowledge distillation for multi-exit networks is proposed, by flowing “knowledge” from the final exit with high accuracy to the early branch exit with lower accuracy. Finally, the effectiveness and efficiency of Branchy-TEE are verified through extensive experiments in real environments, while achieving an optimal balance between performance and hardware resources.

Index Terms—trusted execution environment, early-exit mechanism, security and privacy, knowledge distillation

I. INTRODUCTION

In recent years, Deep Neural Networks (DNNs) have revolutionized the fields of computer vision, autonomous driving, and natural language [1]. This trend has driven the development of DLaaS as a novel computing paradigm, prompting many service providers in specialized fields to deploy their models on top of third-party cloud infrastructure platforms, aiming to provide more convenient and cost-effective services to users worldwide [2].

To address the issue of data security and privacy protection when deploying models in third-party cloud infrastructure platforms, existing cryptography-based approaches can provide sufficient confidentiality and integrity protection for data in transit, computation, and at rest [2][4]. However, complex and frequent cryptographic computations introduce significant computational and communication overheads, which remain impractical for scenarios with high real-time requirements and limited communication bandwidth.

Meanwhile, security protections in any layer of the computing stack can be bypassed by vulnerabilities in the underlying layers, which has driven the need for lowest-layer (hardware-) security solutions. The TEE technologies [5], represented by Intel Software Guard Extensions (SGX) [6], have received increasing attention to ensuring the confidentiality and integrity of data and code by providing a hardware-protected secure area (termed Enclave) [7]. Although SGX provides strong security in an untrusted computing environment, it also suffers from performance degradation under certain conditions. The inference latency has undoubtedly become the most severe obstacle for SGX to develop deep learning systems in the cloud [8][9]. Taegyeong Lee [10] found that deep learning inference inside an Enclave is up to 6.4 times slower than running outside the Enclave. The main reason for the performance degradation is the hardware design limitation. SGX has limited EPC (Enclave Page Cache) memory capacity, which leads to expensive secure page-swapping operations when the memory required by the model exceeds the EPC size [11].

Previous studies have been proposed to address performance degradation. The SLALOM framework proposed by Tramèr [12] delegates all linear layer computations of deep neural networks from TEE to untrustworthy but faster GPUs, resulting in a 6 to 20-fold increase in inference throughput. Similarly, Gu proposed DeepEnclave [13], which attempts to reduce memory usage peaks by dividing the original DNN model into FrontNet (in EPC) and BackNet (in regular memory). However, the above methods cannot guarantee the confidentiality and integrity of the intermediate outputs of the model, resulting in ineffective defense against DNN reconstruction attacks [14]. In conclusion, it remains challenging to achieve secure DNN inference acceleration while efficiently utilizing valuable EPC memory resources [15].

In this paper, we propose Branchy-TEE, a secure and efficient deep learning inference framework for untrusted cloud platforms, to achieve confidentiality and integrity of data and models throughout the inference process. Specifically, Branchy-TEE achieves a balance between security and efficiency under limited EPC resources by building an early-exit network with multiple exits to load different branch networks into the Enclave on demand. Extensive experiments on DNN models of various scales show that Branchy-TEE...
reduces secure inference latency by up to 84.37% compared to native SGX, while introducing a minimum accuracy loss of only 8.39%. In addition, a joint training method of multi-exit networks based on knowledge distillation is proposed, which can compensate the loss of inference accuracy caused by the early exit mechanism to some extent.

II. PROBLEM STATEMENT

A. System Model

The system model of Branchy-TEE is shown in Figure 1 and involves three different entities: a third-party cloud service provider, a model provider, and a data holder.

1) Third-party cloud service provider (CSP): CSP provide powerful computational and storage resources for DNN model inference computation. In addition, CSP supports hardware-protected trusted execution environment (SGX Enclave) for deep learning inference computation on the cloud.

2) Model provider (MP): The MP offloads its model and parameters to the Enclave through a secure communication channel and instantiates its model inside the Enclave and hosts the inference service.

3) Data holder (DH): The DH verifies the inference service is running in the Enclave through remote attestation, and then offloads its private input over a secure communication channel. After the inference service is executed, the output will be returned to the corresponding DP through the established secure channel.

B. Threat Model

With the same trust assumption as the existing research work [16], the only trusted entity in Branchy-TEE is the Intel SGX-enabled CPU on the cloud server, which must support Local/Remote Authentication mechanisms (LA/RA) in addition to providing a trusted execution environment. Apart from that, everything in the infrastructure is untrustworthy, and the potential threats considered in Branchy-TEE are mainly from: Honest-But-Curious cloud providers, malicious co-located cloud tenants, and privileged attackers. Branchy-TEE can preserve confidentiality and integrity for both parties against above potential threats: (1) DH’s private input is not leaked to CSP; (2) The MP’s DNN model (e.g., model architecture and parameters) is not revealed to the CSP and DH in the computation.

III. THE BRANCHY-TEE SYSTEM DESIGN

A. Overall Architecture of Branchy-TEE System

Fig. 1 illustrates an overview of the secure deep learning inference system based on TEE provided by the Branchy-TEE framework.

Phase 1. Initially, the MP instantiates the model inside a real Enclave in a third-party cloud infrastructure via the remote attestation mechanism RA (step ❶). Besides, the model and parameters are offloaded to the Enclave via a secure channel (step ❷).

Phase 2. During the instantiation of the received model, Branchy-TEE loads the first branch network into Enclave 1 via the Layer On-Demand Loading module (LOL). When this early branch fails to make a valid decision, the LOL module loads the second branch network into Enclave 2. It takes the Intermediate Representations (IRs) output from the previous branch network as input to continue the inference computation until the network gives the final inference result. In order to ensure the confidentiality and integrity of IRs transmission, Branchy-TEE implements the authentication among Enclave in the same platform through the Local Attestation Mechanism (LA) provided by SGX (step ❸). After the authentication is passed, a secure TLS communication channel is established to transmit the network intermediate representation IRs (step ❹).

Phase 3. Similar to step ❶ of MP, DH verifies that the inference service is running in the hardware-protected Enclave through the RA mechanism, and establishes a secure communication channel after the verification is passed (step ❹). The DH’s private input is then submitted over this secure channel to the Enclave (step ❼), and once the inference is completed the DH accepts the inference result over the secure channel (step ❽).

B. Layer On-Demand Loading based on Early-exit Mechanism

State-of-the-art SGX-based systems still suffer from significant performance overhead caused by limited EPC memory. Therefore, Branchy-TEE proposes the Layer On-Demand Loading module (LOL) based on early-exit Mechanism to load different branching networks layer by layer in a sequential manner, aiming to expect the model to make inference decisions at an early stage and minimize the EPC memory usage.

1) Relevant Definitions: The network structure of DNN inference tasks is hierarchically structured. We can define DNN inference formally as a function: \( y = f^*(x) = f_{\theta_n} f_{\theta_{n-1}} \cdots f_{\theta_1}(x) \). The input \( x \) is mapped to the output \( y \) by a layer-by-layer nonlinear transformation, where \( f_{\theta_i} \) and \( \theta_i \) denote the \( i_{th} \) layer subfunction and its corresponding parameters, respectively, and \( n \) is the number of the DNN layers. The intermediate representation of the \( i_{th} \) hidden layer can be denoted as \( IR_i = f_{\theta_i} f_{\theta_{i-1}} \cdots f_{\theta_1}(x) \). Thus, we can give a formal definition of early-exit network with multiple exits and other related definitions as follows:
Definition 1. (Early-Exit Network) Given a DNN model with $|\mathcal{M}|$ branch exits $y = f^M(x, \theta) : \mathcal{X} \to \mathbb{P}^{|\mathcal{M}|}$, where $\mathcal{M} = \{e_1, e_2, \cdots, e_{|\mathcal{M}|}\}$ denotes the layer of the backbone network where the branch exit is located. For any branch exit $\mathcal{M}_i$, its classification probability distribution is calculated $P_{\mathcal{M}_i} : \mathbb{R} \to \Delta^K$ as follows:

$$P_{\mathcal{M}_i} = c_{\mathcal{M}_i}(IR_{e_i}, \bar{\theta}_{\mathcal{M}_i})$$  \hspace{1cm} (1)

where $c_{\mathcal{M}_i}$ denotes the layer of the backbone network where the branch exit $\mathcal{M}_i$ is located, $c_{\mathcal{M}_i}$ and $\bar{\theta}_{\mathcal{M}_i}$ denote the branch classifier and the corresponding parameters, respectively, and $\Delta^K$ denotes the probability distribution of the model output $y$ corresponding to $K$ classes.

Definition 2. (Branch decision confidence) For any current branch exit $\mathcal{M}_i$, the output probability distribution of that branch classifier $c_{\mathcal{M}_i}$ is $P_{\mathcal{M}_i}$, and the confidence degree $\mathcal{H}(\cdot) \in (0, 1]$ of that branch decision can be measured by calculating the normalized entropy of $P_{\mathcal{M}_i}$, as follows:

$$\mathcal{H}(P_{\mathcal{M}_i}) = -\frac{1}{\log K} \sum_{i=1}^{K} p_{\mathcal{M}_i} \log p_{\mathcal{M}_i}$$  \hspace{1cm} (2)

When $\mathcal{H}(P_{\mathcal{M}_i}) < \beta_i$, it means that the branch is confident enough to perform an early exit on the network, otherwise the inference continues to the next branch execution, where $p_{\mathcal{M}_i} \in P_{\mathcal{M}_i}$ and $\beta_i$ is the pre-set decision threshold of the current branch.

Branchy-TEE is mainly oriented towards feed-forward DNN for classification tasks, where the inference process is performed by extracting feature representations (i.e., IR) layer by layer. Therefore, the computation of each layer is semantically independent, and the IR is dynamically generated by each layer and is only accessed by the layer that generated it and the connected next layer in a short time. It leads to very low reusability of EPC memory and triggers a high frequency of secure paging in EPC. This feature motivates Branchy-TEE to cut the original network into $M$ independent mutually exclusive branch networks loaded into the secure Enclave on demand.

2) Secure Inference Acceleration Algorithm in Enclave: In this subsection, we first detail the security inference acceleration algorithm, termed FINE (Fast Inference IN Eclaves) in Branchy-TEE, as shown in Algorithm 1 below. In summary, based on the early-exit mechanism, the FINE algorithm is built on the BranchyNet[17]. After decrypting the user’s sensitive input data within the first Enclave (line 4 of the algorithm), any Enclave, provides a secure computation for its associated branch network $\text{branch}_{\mathcal{M}_i}$. The $\text{branch}_{\mathcal{M}_i}$ is consisted of the backbone network $f^{M}_{\theta_{e_{i}}^{1}} \cdots f^{M}_{\theta_{e_{i+1}-1}}$ (line 7 of the algorithm) and the branch classifier $c_{\mathcal{M}_i}$ (line 8 of the algorithm) in series. When $\mathcal{H}(c_{\mathcal{M}_i}(IR_{e_i}, \bar{\theta}_{\mathcal{M}_i})) < \beta_i$, Branch-TEE supports network early exit and the final inference result is given by this branch, and the output of $\mathcal{M}_i$ will be sent to the DH via a secure channel (lines 12 to 14 of the algorithm). The confidence degree of the current branch network decision can be measured by calculating the entropy of the inference result (soft-label) (as shown in definition 2). If $\mathcal{H}(c_{\mathcal{M}_i}(IR_{e_i}, \bar{\theta}_{\mathcal{M}_i})) > \beta_i$, the intermediate representation of the backbone network $IR_{e_i}$ will be sent as input to $\text{Enclave}_{\mathcal{M}_{i+1}}$ via a secure channel (line 16 of the algorithm), where the following branch network is located, to continue the inference task layer by layer.

Algorithm 1 FINE: Fast Inference IN Eclaves

Input: $\bar{x}$, encrypted user input; $f^{M}_{\theta}$, a DNN model with $|\mathcal{M}|$ branch exits; $\mathcal{M}$, branch exit points; $\theta$, parameters of each layer; $\bar{\theta}$, corresponding parameters of branch classifier $c_{\mathcal{M}_i}$; $B = \{\beta_1, \beta_2, \cdots, \beta_{|\mathcal{M}|}\}$, set of branch exit decision thresholds; $K$, number of categories in the model output.

Output: $\hat{y}$, encrypted inference result

1: for $i = 1$ to $|\mathcal{M}|$ do
2: Initialize decision confidence $\text{confidence} = 0$
3: if $i == 1$ then
4: $x \leftarrow \text{Decryption}(\bar{x})$
5: $IR_{e_1} = f^{M}_{\theta_{e_{1}}} \cdots f^{M}_{\theta_{e_{i}}} (x)$
6: else
7: $IR_{e_i} = f^{M}_{\theta_{e_{i}}} \cdots f^{M}_{\theta_{e_{i+1}-1}} (IR_{e_{i-1}})$
8: $P_{\mathcal{M}_i} = c_{\mathcal{M}_i}(IR_{e_i}, \bar{\theta}_{\mathcal{M}_i})$
9: for $p_{\mathcal{M}_i}$ in $P_{\mathcal{M}_i}$ do
10: $\text{confidence} = \text{confidence} + p_{\mathcal{M}_i} \log p_{\mathcal{M}_i}$
11: if $\text{confidence} < \beta_i$ or $i == |\mathcal{M}|$ then
12: $\hat{y} \leftarrow \text{Encryption}(\text{arg max } P_{\mathcal{M}_i})$
13: return $\hat{y}$
14: end if
15: end for
16: $\text{Enclave}_{\mathcal{M}_{i+1}} \leftarrow \text{Sending}(\text{Enclave}_{\mathcal{M}_i}, IR_{e_i})$
17: continue

C. Joint Training based on Knowledge Distillation for Multi-exit Networks

In order to compensate for the loss of inference accuracy introduced by the early exit mechanism, a joint training method based on knowledge distillation is proposed to transfer “knowledge” from the last exit (teacher network) with high classification accuracy to the early branch exit (student network) with lower accuracy.

As shown in Fig. 2, the loss function of the knowledge distillation network (illustrated in Eq. 3) consists of the distillation loss extracted from the Soft-target by the teacher network and the original classification loss of the student network (Hard-target), both jointly weighted. The “distillation” method incorporates a “temperature” parameter $T$ into the softmax function, causing the output of the softmax layer to become smoother as $T$ increases. As a result, the information carried by negative labels is amplified relative to positive labels, and the model training focuses more on negative labels [18]. Specifically, given a training set $\{(x_n, y_n)\}_{n=1}^N$, the overall loss of the joint training is calculated as follows:
\[ L_{\text{total}} = \frac{1}{N} \sum_{n=1}^{N} \left[ L_{\text{class}} (x_n, y_n) + \lambda L_{\text{dist}} (x_n) \right] \] (3)

where \( L_{\text{class}} \) is the classification loss of the multi-exit network (dashed arrow in Figure 2), \( L_{\text{dist}} \) is the distillation loss (solid arrow in Figure 2), and \( \lambda \) is the superparameter that balances between the two classes of losses.

The distillation loss \( L_{\text{dist}} \) is utilized to transfer knowledge to each of the early exits \( s_{e_i} \), improving the prediction accuracy of the early branches. Specifically, the last exit is used as the teacher network \( t \), and the cross-entropy between the \( \text{softmax} \) distribution of the branch network and the \( \text{softmax} \) distribution of the teacher network is calculated as the branch distillation loss \( L_{t \rightarrow s_{e_i}} \) under the temperature \( T = \tau \) condition. The losses of each branch are then weighted and summed to obtain the overall distillation loss \( L_{\text{dist}} \).

\[ L_{\text{dist}} = \sum_{i=1}^{[M-1]} L_{t \rightarrow s_{e_i}} = \sum_{i=1}^{[M-1]} \left[ -\sum_{k=1}^{K} p_k^i \log (q_k^{e_i, \tau}) \right] \]

where \( p_k^i \) denotes the probability of the \( \text{softmax} \) output of the teacher network \( t \) at temperature \( \tau \) on the \( i \)-th class, \( v_i \) denotes the logits of the teacher network output, \( q_k^{e_i, \tau} \) denotes the probability of the \( \text{softmax} \) output of the branch network \( e_i \) at temperature \( \tau \) on the \( i \)-th class, and \( z_k^{e_i} \) denotes the logits of the branch network output.

As the teacher network may also have a certain error rate, equation (3) introduces a classification loss \( L_{\text{class}} \) to mitigate the propagation of errors from the teacher model to the student model by using the ground truth label. The classification loss is defined as follows:

\[ L_{\text{class}} = \sum_{i=1}^{[M]} \left[ -\frac{1}{\tau^2} \sum_{k=1}^{K} c_k \log q_k^{e_i, T=1} \right], \]

where \( c_k \in \{0, 1\} \) denotes the ground-truth value on the \( k \)-th class and \( q_k^{e_i, \tau=1} \) denotes the \( \text{softmax} \) calculation for each branch without distillation (\( T = 1 \)). Since the introduction of temperature \( \tau \) into the \( \text{softmax} \) function in the \( L_{\text{dist}} \) to soften the probability distribution, the size of the gradient generated by the distillation loss is reduced by \( \tau^2 \) compared to the categorical loss, we multiply the coefficients of \( \frac{1}{\tau^2} \) in Eq. (5) to ensure that the distillation loss and the classification loss contribute essentially the same amount of gradient.

**IV. EXPERIMENTS**

**A. Experiment Setup**

Environment. Branchy-TEE is implemented based on the SGX Library Operating System Graphene-sgx[19], with a higher degree of compatibility. The processor supports SGX feature and has 128MB of EPC memory.

Models and datasets. We employ DNN models of different scales to evaluate Branchy-TEE: including AlexNet, VGG19, MobileNet v1, ResNet50, ResNet101, and ResNet152. Regarding the dataset, we used the CIFAR-10 [20] to compare and validate the performance of our proposed secure inference algorithm.

Baseline. Comparison with Branchy-TEE by the following baseline methods: (1) Native-DNN, DNN inference tasks are running in unprotected memory outside of Enclaves; (2) Native-SGX, DNN inference tasks are forced to be executed in the native SGX Enclave without providing any optimization mechanism.

**B. Result Analysis**

First, the performance overhead of inference in and outside the Enclave for DNN models of different scales is illustrated in Table I. It is evident from the table that the inference latency of Native-SGX is an order of magnitude higher than that of Native-DNN, with an average latency of 33x. As mentioned earlier, the reason for such severe performance degradation is that all models run with a peak memory much larger than the upper limit of EPC memory, triggering a large number of EPC Secure Paging operations, which significantly reduces the inference speed of the models within the Enclave.

The conclusions drawn in Fig. 3 reveal that the more backward branching network has a higher demand for EPC memory, and the cost of an EPC page swap is quite expensive (up to 40K cycles) [21]. It motivates us to assign a more loose branch decision threshold \( B \) in Algorithm 1 to encourage the network to give a final decision at an early stage.

Furthermore, Table II compares the secure inference performance between Branchy-TEE and Native-SGX in terms of...
TABLE I: Performance overhead introduced by executing deep inference tasks in Native-SGX.

<table>
<thead>
<tr>
<th>models</th>
<th>model sizes</th>
<th>Peak memory</th>
<th>Inference Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Native-DNN</td>
</tr>
<tr>
<td>AlexNet</td>
<td>492KB</td>
<td>161MB</td>
<td>0.022s</td>
</tr>
<tr>
<td>VGG19</td>
<td>77MB</td>
<td>335MB</td>
<td>0.146s</td>
</tr>
<tr>
<td>MobileNet v1</td>
<td>13MB</td>
<td>190MB</td>
<td>0.0069s</td>
</tr>
<tr>
<td>ResNet50</td>
<td>91MB</td>
<td>339MB</td>
<td>0.287s</td>
</tr>
<tr>
<td>ResNet101</td>
<td>163MB</td>
<td>484MB</td>
<td>0.472s</td>
</tr>
<tr>
<td>ResNet152</td>
<td>223MB</td>
<td>618MB</td>
<td>0.694s</td>
</tr>
</tbody>
</table>

Fig. 3: The performance of each early-exit branch of different model running separately in Enclave.

average peak memory usage and average inference latency, respectively. Overall, Branchy-TEE significantly outperforms Native-SGX for all target models. For the AlexNet model with the slightest performance improvement, the average peak memory requirement of Branchy-TEE decreases by 16.28% compared to Native-SGX. In contrast, for ResNet152, with the highest EPC memory requirement, this decrease is as high as 83.06%. Correspondingly, the average inference latency generated by the AlexNet is reduced by up to 33.10% compared to Native-SGX, and up to 84.15% in ResNet152. However, the model accuracy loss introduced by the early-exit mechanism is not negligible, ranging from 8.39% for VGG19 to 12.46% for ResNet-101, respectively.

TABLE II: Security performance comparison between Branchy-TEE (ours), Native-SGX, and Native-DNN.

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy loss (Native-DNN)</th>
<th>Average peak memory</th>
<th>Average inference latency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ours</td>
<td>Native-SGX</td>
<td>Ours</td>
</tr>
<tr>
<td>AlexNet</td>
<td>9.15%</td>
<td>1.08s</td>
<td>33.10%</td>
</tr>
<tr>
<td>VGG19</td>
<td>8.39%</td>
<td>0.66s</td>
<td>84.37%</td>
</tr>
<tr>
<td>MobileNet v1</td>
<td>10.87%</td>
<td>1.04s</td>
<td>82.23%</td>
</tr>
<tr>
<td>ResNet50</td>
<td>11.13%</td>
<td>1.18s</td>
<td>79.88%</td>
</tr>
<tr>
<td>ResNet101</td>
<td>12.46%</td>
<td>1.18s</td>
<td>79.88%</td>
</tr>
<tr>
<td>ResNet152</td>
<td>11.07%</td>
<td>2.95s</td>
<td>84.15%</td>
</tr>
</tbody>
</table>

Fig. 4: Performance impact of knowledge distillation on the accuracy of each branchy with different temperature.

Fortunately, the joint training method based on knowledge distillation mentioned above is able to compensate to some extent for the accuracy loss introduced by the multiple exit early retirement network. As shown in Fig. 4 different branching networks show a certain degree of upward trend in Top-1 accuracy after knowledge distillation. In addition, different distillation temperatures have different effects on different models and branches. For some simple models, such as AlexNet and Vgg19, the accuracy improvement effect of distillation temperature $T = 4$ is not better than that at $T = 2$. Even at the distillation temperature $T = 8$, the accuracy of AlexNet and VGG19 is slightly lower than that of the architecture without knowledge distillation.

Conversely, for the ResNet50 network, the accuracy improvement for the network showed an increasing trend with increasing distillation temperature. In general, the choice of distillation temperature $T$ has a certain relationship with the size of the model. The model with a relatively small number of parameters cannot learn all the knowledge of the teacher
network, and a relatively low temperature cannot amplify the information of the soft-target, but can effectively ignore the noise generated by some negative labels.

V. CONCLUSIONS

In this paper, we propose a secure and efficient inference framework for deep learning in untrustworthy cloud platforms, termed Branchy-TEE, which aims to protect the confidentiality and integrity of data and models throughout the inference process using SGX. Moreover, Branchy-TEE dynamically loads the inference network into the TEE on-demand based on early-exit mechanism, breaking the hardware performance bottleneck of SGX. Moreover, a joint training method based on knowledge distillation for multi-exit networks is proposed to compensate for the loss of inference accuracy introduced by the early exit mechanism.

REFERENCES


HaeNAS: Hardware-Aware Efficient Neural Architecture Search via Zero-Cost Proxy

Yaodanjun Ren†, Chen Chen†, Zhengwei Qi†
†Shanghai Jiao Tong University
§State Key Laboratory of Mathematical Engineering and Advanced Computing

Abstract
The practical use of advanced DNN models is hindered by limited hardware resources and high computation demands. Neural Architecture Search (NAS) is becoming a default technique which automatically discovers architectures that are competitive with handcraft ones. However, existing methods only prioritize accuracy and overlook hardware-related factors. To address this, we introduce HaeNAS (Hardware-aware efficient NAS), which considers both accuracy and computation cost on specific hardware platforms. The search space of HaeNAS consists of several stages, each allowing different convolution kernels, layer numbers, layer widths, and operation types. We use a data-driven approach to predict the latency and energy consumption on target hardware, and we improve the zero-cost proxy based on network pruning research to speed up the NAS process. With these techniques, HaeNAS finds a target network within 160 GPU hours, which achieves 80.7% top-1 accuracy on the ImageNet, with a latency of 10.4ms and energy consumption of 931mJ.

Keywords: neural architecture search, convolutional neural network, evolution algorithm, neural network acceleration

1 Introduction
Deep neural network models have achieved impressive results in various applications [8, 11]. However, running AI applications on specific hardware must not only achieve high accuracy but also meet latency and energy constraints. Thus, the high computation demands required by high accuracy hinder the practical application of DNN models in real-world scenarios, especially on embedded or edge devices.

Many recent studies have attempted to improve the efficiency of CNN models by utilizing software-related or hardware-related optimization methods [13]. In the hardware domain, a popular approach is to leverage the specific hardware characteristics of hardware devices such as GPUs, FPGAs, and NPUs to enhance the inference and training speed of DNNs. However, for a given DNN model, hardware optimization is constrained by the computation and storage resources inherent in the hardware itself. Optimization solutions on the software side have also received attention from researchers, such as model pruning and quantization. For example, network pruning methods usually reduce the parameter size by identifying a criterion of significance for each component (i.e., parameter, filter, layer), such as the magnitude of the weight [11]. Recently, pruning-at-initialization methods [4, 5] have attracted a lot of interest as they can reduce the training cost while achieve similar accuracy.

Neural Architecture Search (NAS) has become a popular approach for automatically discovering competitive neural network architectures, surpassing those designed by humans [2]. Existing methods [1, 12] primarily focus on accuracy while neglecting other hardware-related factors, such as latency, energy, and memory. However, the optimality of DNN models depends on both the model architecture and target devices. To address this, we propose a hardware-aware NAS method, aiming to find the Pareto-optimal architectures within the search space. There are several challenges. Firstly, different platforms require different operations, and efficiency metrics are not simple functions of MACs (multiply–accumulate operations). Therefore, it is necessary to customize the search space and strategy for a specific hardware platform. Secondly, customizing search options for hardware will inevitably increase the complexity of the search space, rendering the search cost more difficult to bear.

Inspired by the above observations, we propose a NAS method to find a resource-efficient network on the target hardware. As illustrated in figure 1, our hardware-aware efficient NAS (HaeNAS) consists of three steps. First, we design a flexible search space that takes into account the hardware features. Second, we employ zero-cost proxies to implement hardware-aware evolution search. Finally, post processing, such as compound scaling and adding SE block, are utilized. Our contributions are as follows:
We propose a lightweight NAS methodology which can quickly explore a wide search space, considering resource constraints including latency and energy.

- We designed a flexible search space that allows for selection among different depths, widths, convolution kernels, and operation types in each stage.
- We evaluate models with direct hardware metrics such as latency and energy consumption, instead of proxy metrics such as model size and FLOPs.
- We improve the pruning-at-initialization methods which only use a minibatch of data, leading to a significant reduction in the search cost.

## 2 Methodology

In this section, we present the technical details of HaeNAS, which searches efficient CNN models for specific hardware platforms in a data-driven way.

### 2.1 Problem formulation

Our goal is to find CNN models with both high accuracy and low resource consumption. Resource consumption, such as latency and energy, during model inference is critical, as they can directly affect the user experience and the number of model executions. Previous NAS approaches often optimize for indirect metrics, such as FLOPS, MACs, and model sizes. We consider inference latency and energy consumption by running and tracking CNN models on the target devices and incorporating these hardware metrics into our objective function. The NAS problem is formulated as

\[
\max_{a \in \mathcal{A}} \text{Score}(a) \times \left[ \frac{\text{LAT}(a)}{L} \right]^{\lambda} \times \left[ \frac{\text{EC}(a)}{E} \right]^{\omega} \tag{1}
\]

where \(\text{Score}(a)\) measures the accuracy of architecture \(a\), \(\lambda\) and \(\omega\) respectively represent the importance of latency and energy consumption. \(\lambda\) could be defined as equation 2. And \(\omega\) is defined in a similar way.

\[
\omega = \begin{cases} 
\alpha, & \text{if LAT}(a) \leq L \\
\beta, & \text{otherwise} \end{cases}
\]

where \(\alpha\) and \(\beta\) are hyperparameters.

Given an architecture search space \(\mathcal{A}\), HaeNAS aims to find an optimal architecture \(a \in \mathcal{A}\) that after training it can achieve the optimal accuracy and resource consumption trade-off. Therefore, in our work, we focus on three factors of the problem: (1) the architecture search space \(\mathcal{A}\), (2) the objective function that considers latency and energy consumption, and (3) a lightweight search algorithm that scores architectures before training.

### 2.2 Search space design

The effectiveness of NAS algorithms is directly dependent on search space design. Ideally, it should encompass numerous excellent candidate architectures while avoiding unnecessary complexity. Predefined macro architecture and layer-level search strategy have been shown to reduce search cost without compromising model performance [9]. NASNet [15], for instance, improved search efficiency by 7 times and achieved higher accuracy compared with the previous work [14] by employing suitable macro architecture design.

HaeNAS adopts a macro architecture based on the MobileNets [12] and searches in the level of stage. As shown in Table 1, HaeNAS’s search space is divided to 9 stages, where the first and last stages are fixed. The first stage contains a regular convolution layer with a 3x3 kernel and stride of 2. The last stage contains an average pooling layer and two convolution layers, which is proposed by MobileNetv3 [12] to save cost without sacrificing accuracy. The middle 7 stages are searchable, while the input resolution and stride of each stage is fixed. HaeNAS searches the setting of stages, including operation type, kernel size, stage width, and stage depth.

The inverted bottleneck (IBN) block with DWConv, which is proposed by MobileNets [12], is widely. It has fewer parameters and MACs, making it more suitable for mobile devices. However, the theoretical computational complexity (e.g. MACs or FLOPs) does not often necessarily correspond to inference speed. The inference latency may be proportional to the MACs of the model on one platform, while being proportional to the memory access on another platform. Therefore, when designing a model, it is crucial to select a model that aligns with the platform’s features.

Many current NAS method, such as Once-for-all and MnasNet, only consider IBN block when designing search spaces [1]. This severely limits the flexibility of the resulting models and fails to provide models that are better suited to hardware platform. To address this, we propose searching for convolution operation type in addition to kernel size, stage depth, and width. As shown in Fig 2, besides IBN block, the search space of HaeNAS also includes fused inverted bottleneck layers (Fused-IBN). The Fused-IBN block replaces the combination of DWConv and pointwise Conv layers in IBN block with a regular Conv. Although this increases the computation cost, it reduces the I/O overhead of intermediate results.

### 2.3 Hardware aware evolution search

HaeNAS improves the aging evolution (AE) algorithm proposed by AmoebaNet [7], and incorporates hardware awareness in fitness evaluation. The procedure of Hardware-Aware Aging Evolution (HAE) algorithm is shown in Algorithm 1.
Algorithm 1: Hardware-aware aging evolution

Output: candidate with the highest fitness
1. population ← empty queue; \( \triangleright \) current population
2. history ← \( \varnothing \); \( \triangleright \) save all the excellent models
3. while \( \text{population} < P \) do
   4. model.arch ← RandArch();
   5. model.fit ← HaeFitness(model.arch);
   6. add model to population queue;
   7. add model to history set;
4. end
5. while \( \text{history} < C \) do
   6. sample ← \( \varnothing \); \( \triangleright \) candidate parents
   7. while \( \text{sample} < S \) do
      8. candidate ← random member from population;
      9. add candidate to sample;
   10. end
   11. parent ← highest-fitness member in sample;
   12. children.arch ← HaeMutation(parent.arch);
   13. children.fit ← HaeFitness(children.arch);
   14. add highest-fitness child to history;
   15. add highest-fitness child to population;
   16. remove the oldest from population;
5. end

We use queue to save population participating in evolution and use set to save all excellent individuals in the history. HAE algorithm adds young individuals from the right and removes old individuals from the left of the queue based on the FIFO principle. The history set only stores the model architecture without saving weights, thereby avoiding occupying too much memory. HAE inherits the “age” concept proposed by AE [7] to ensure a regular elimination and update of the population. Individuals with the longest existence time, rather than those with the worst performance, will be eliminated. This approach avoids the situation where elite individuals produce a large number of offspring, which would compromise the diversity of the population. Furthermore, the HAE algorithm prevents being trapped in local optima by assigning a probability for poor individuals to generate offspring through random selecting candidate parents. The HAE algorithm also maintains a faster convergence rate by preserving a subset of superior individuals in the population.

Some work [1, 13] achieves hardware awareness by including latency as part of the loss function, while HaeNAS achieves it more directly by incorporating hardware metrics into the fitness function of the evolution algorithm. As shown in equation 1, we adopt a custom weighted product approach to evaluate fitness by aligning accuracy, latency, and energy consumption metrics.

To evaluate the fitness of candidate individuals, it is necessary to obtain their accuracy scores. In AmoebaNet [7], candidate models are trained for 50 epochs under the same setting to obtain their accuracy on CIFAR-10. However, it is very expensive to train a model on ImageNet, which is the target dataset of our HaeNAS. To reduce the training cost, some NAS algorithms, such as ProxylessNAS [1], adopt methods like training supernet and inheriting supernet weights for subnet. However, these methods still require significant computation cost. Inspired by pruning-at-initialization methods [4], HaeNAS proposes to rank candidate networks using zero-cost metrics instead of accuracy. While accuracy is obtained after tens of epochs of training, the score of the zero-cost metric is obtained by a single forward/backward propagation, thus saving the search cost.

We improve the following pruning-at-initialization methods, extending them to scoring the entire candidate network. These metrics were previously used at the granularity of a single neuron (e.g. a parameter or a channel), now we adapt them by sum up every neuron’s score to get the score of entire model. Here, \( L \) represents the loss function of a candidate model with weight parameters \( \theta \). \( \odot \) represents the Hadamard product, which refers to the element-wise multiplication of two matrices with the same dimensions. The grad_norm means we sum the Euclidean norm (L2) of the gradients after a single minibatch. The snip [4] is proposed by Lee et al, which measures the change in loss function when a specific operation or parameter is removed. The grasp [10] is similar to snip, while it measures the change in gradient norm instead of loss function. The jacob_conv [5] is proposed by Mellor et al, which captures the correlations between activations in the network when presented with different inputs. A lower correlation is indicative of better performance in distinguishing between different input classes.

\[
S_{\text{grad.mnorm}} = \sqrt{\sum \left( \frac{\partial L}{\partial \theta} \right)^2} \tag{3}
\]
\[
S_{\text{snip}}(\theta) = \frac{\partial L}{\partial \theta} \odot \theta \tag{4}
\]
\[
S_{\text{grasp}}(\theta) = -\left( \frac{\partial L}{\partial \theta} \right) \odot \theta \tag{5}
\]
\[
S_{\text{jacob.conv}} = -\sum_{i=1}^{N} \left[ \log (\sigma_{j,i} + k) + (\sigma_{j,i} + k)^{-1} \right] \tag{6}
\]
Another issue is the value of $\lambda$, $\omega$, $L$ (i.e. target inference latency), $E$ (i.e. target energy consumption) in the fitness function. Figure 3 presents the fitness curve under two different sets of $(\alpha, \beta)$ values. For illustration purposes, the inference latency $T$ is set to 80ms and energy consumption is not considered. The upper curve applies a soft constraint on the target inference latency with $(\alpha = -0.05, \beta = -0.05)$, while the lower curve imposes a hard constraint with $(\alpha = -0, \beta = -0.7)$. A hard constraint can prevent the model from violating the inference latency constraint by rapidly decreasing the fitness value, but the fitness score would depend solely on the accuracy when the latency constraint is not exceeded. A smoother adjustment of the fitness function value is more beneficial for balancing accuracy and hardware metrics. Therefore, the HaeNAS algorithm adopts a soft-constrained fitness function.

An empirical rule for choosing $\lambda$ and $\omega$ is to ensure that the Pareto-optimal solutions have similar fitness value. In the EfficientNets [6], doubling the latency typically comes with a 3% relative accuracy improvement. Specifically, given two models $m_1$ and $m_2$ with the same energy cost, and considering only inference latency and accuracy, if model $m_1$ has an inference latency of $l$ and an accuracy score of $s$, while model $m_2$ has an inference latency of $2l$ and an accuracy score of $(1+3\%)s$, the fitness value for them should be roughly equal. Thus, the value of $\alpha$ is approximately -0.05. The situation is similar for energy cost. Therefore, in the subsequent experiments, unless otherwise specified, the HaeNAS algorithm sets the scaling factors for latency and energy to -0.05.

![Figure 3](image)

**Figure 3.** Objective function defined by equation 1

### 3.1 Experimental setups

**Datasets & Evaluation Metrics.** We evaluate HaeNAS on image classification task. We use ImageNet-1K dataset, which consists of 1.2 million training images and 50,000 validation images, covering 1000 categories. In addition to the classification accuracy, HaeNAS evaluates the inference latency on Xeon CPU and GTX 3080 GPU, and the inference energy consumption on GTX 3080 GPU and Raspberry Pi 4.

**Hardware Platforms.** We conduct experiments on three different hardware platforms. The first one is a Xeon E5-2650 CPU, which is a server CPU. As it is in a stable power supply environment, we only focus on its inference latency. The second one is a GTX 3080 GPU, which is a high-performance GPU for both training and inference. We use the nvidia-smi utility to measure its energy consumption. The third one is a Raspberry Pi 4, which is a widely used edge development board. We use an FNIRSI-FNB58 power meter to track its voltage and current, enabling us to obtain power consumption information.

**Latency and Energy Prediction Models.** HaeNAS employs multi-layer perceptron (MLP) models to predict inference latency and energy consumption. To train the prediction models, 1000 random models are sampled from the search space. To avoid the impact of the DNN model’s computation startup and the randomness in the computation process, we adopt the approach of warm-up and multiple runs to obtain an average value. Meanwhile, different devices have different computing parallelism, and appropriate loads need to be set when measuring hardware metrics, which is achieved by selecting an appropriate batch size.

### 2.4 Post processing

Modern series models such as MobileNets [12] and EfficientNets [9] typically include multiple models that vary in size and accuracy to meet different resource requirements. HaeNAS intentionally imposes stricter constraints on the target inference latency $T$ and target energy consumption $E$ in the fitness function. We employ compound scaling [9] to simultaneously increase the model’s depth, width, and input resolution. Compound scaling achieves higher precision return than scaling a single factor alone by allocating scaling coefficients for each factor in a balanced way. We perform a grid search to find scaling factors for depth, width, and input resolution under different multiples of computational cost. Additionally, the proposed method employs manual adjustments. Firstly, it restricts the maximum input resolution to 380 to avoid excessive training and inference costs. Secondly, more network layers are added in the later stages, as aggressive expansion of depth is unnecessary in the early stages.

To enhance the accuracy, HaeNAS further incorporates a lightweight attention module: squeeze-and-excitation(SE) block [3]. It applies attention to channels. Given an input feature map, the SE block first performs global average pooling for each channel. Then, the squeeze operation captures channel correlations through two cascaded fully connected layers. Finally, channel-wise multiplication is performed between the activation values and the input feature map. To avoid introducing too much computation cost, we selectively remove unimportant SE blocks. The standard deviation of the activation values across different images on each channel is calculated. If the standard deviation is small, it indicates that the SE block does not help distinguish which channel is more important, so the SE block can be removed.
3.2 Main results

Train-free proxy for lightweight NAS. The effectiveness of adopting zero-cost metrics as the evaluation strategy of HaeNAS needs to be confirmed. We use NAS-Bench-201 as the benchmark model set and evaluate the effectiveness with Spearman’s rank correlation coefficient. The Spearman’s rank correlation coefficient measures the order relationship between variables, rather than the numerical relationship.

The NAS-Bench-201 provides accuracy information for three datasets: CIFAR-10, CIFAR-100, and ImageNet 16-120. Figure 4 presents the performance results of zero-cost proxies. In terms of a single criterion, jacob_conv achieves the highest score. It captures the model’s ability to differentiate between different inputs, i.e., classification, based on activation values, and thus has considerable potential. Following closely behind are grad_norm and snip, both of which involve gradient values, with Spearman’s correlation coefficients around 0.6. The calculation of loss and the update of the parameters result in the loss being called gradient propagation. Thus, the gradient value to some extent reflects the model’s learning direction and learning ability.

Due to the fact that the selected evaluation metrics have different focuses, HaeNAS utilizes a majority voting scheme to decide which individuals to keep during each evolution round. Specifically, jacob_conv, grad_norm, and snip are jointly considered to determine the candidates to be preserved. Moreover, for recording fitness scores and selecting potential parents, the value of jacob_conv is used.

Results under different computational resource constraints. The searching results of HaeNAS for base models on different platforms are shown in Figures 5. The letters denote the corresponding platform, with "G" representing GPU, "C" representing CPU, and "E" representing edge device Raspberry Pi 4. Different colors indicate different convolution kernel sizes, and F-IBN and IBN are used to distinguish operation types. The number inside the rectangle represents the expansion ratio, and the number outside the rectangle represents the stage depth.

Different hardware platforms result in architectures with distinct characteristics when searched by HaeNAS. For instance, HaeNAS-G tends to choose less layers and F-IBN block, with a preference for large kernel and big expansion ratio in the early stages. HaeNAS-C tends to choose more layers and smaller expansion ratio. Meanwhile, HaeNAS-E tends to select IBN block, which result in less computation cost. Overall, all three models prefer larger kernel sizes and retain SE block in the later stages.

Accuracy and search cost. Table 2 presents the accuracy result of HaeNAS and other mainstream models, both handcraft or automatically designed. Compared to MobileNetv2, HaeNAS models have improved accuracy on the ImageNet dataset by 4.6% to 6.3% while maintaining similar inference time.

Compared to other autoML methods, the HaeNAS based on zero-cost metrics has a significant advantage in search cost. Compared to reinforcement learning methods, the required search cost of HaeNAS is reduced by nearly 300 times. Even compared to the ProxylessNAS based on gradient and super network parameter sharing, HaeNAS achieves 1.25 times computation cost savings. These benefits are mainly due to the use of zero-cost proxy metrics that reduce unnecessary additional training.

Hardware metrics. HaeNAS applies compound scaling to the base models obtained by the hardware-aware evolutionary algorithm. Figures 6 shows the accuracy-latency comparisons of the HaeNAS-C and HaeNAS-G models after scaling, with M/L denoting the scaled models. HaeNAS-C-L can achieve 80.1% Top-1 accuracy at a speed of 137ms on the CPU, and the latency is 86% of that of EfficientNet at the same accuracy. On 3080 GPU, the inference latency of HaeNAS-G models are only 44% to 49% of that of EfficientNets at the same accuracy. We believe this is due to the inclusion of F-IBN block in the search space, which is more conducive to the parallel performance of GPU-like hardware.
We present HaeNAS, a hardware-aware efficient neural architecture search framework. It includes three steps. First, it designs the search space based on hardware characteristics and predefines the macro architecture of models. Second, in order to save search cost, HaeNAS proposes to use zero cost and hardware-aware evolution algorithm to search candidates for target devices. The actual target device latency and energy consumption of sampled networks are used to train prediction models. Third, compound scaling and selective SE block are adopted to find models that strike a balance between accuracy and hardware metrics. The accuracy of the models found by HaeNAS on ImageNet can be comparable to mainstream models, both handcraft and automatically designed. The latency and energy consumption comparison on all target hardware platforms (Xeon CPU, 3080 GPU, and Raspberry Pi 4) confirms the effectiveness of the proposed HaeNAS methodology.

Acknowledgments
This work was supported by the Open Project Program of the State Key Laboratory of the Mathematical Engineering and Advanced Computing (2020A10), the National NSF of China (NO. 62141218), and Shanghai Key Laboratory of Scalable Computing and Systems.

References
Abstract—Probabilistic timed automata (PTAs) allow for analyzing systems which operate under the existence of both probability and time. Probabilistic model checking for PTAs requires formal models as a basis. In this paper, we consider a learning algorithm for PTAs with one clock (OPTAs) which extends Angluin’s L algorithm. For the black-box systems, we evaluate the probabilities by sampling system traces, and apply the framework of Probably Approximately Correct (PAC) learning to provide correctness guarantees respect to error and confidence parameters. Experiments with the implementation of our learning algorithm include the sender of CSMA/CD protocol and randomly generated examples. Evaluating results show the effectiveness of our learning algorithm.

Index Terms—model inference, active automata learning, probabilistic timed automata

I. INTRODUCTION

Model learning is an efficient technique where we are able to get formal models for black-box systems. It enables the formal techniques, such as model checking, for systems. Passive learning and active learning are different two strategies in model learning. Specifically, passive learning outputs the learned models through before-hand data, but the data active learning uses grow gradually by asking for the systems under learning (SUL) during learning. Increased active learning algorithms are derived from the Angluin’s L algorithm [1] which learns deterministic finite automata (DFAs), e.g. learning other automata variants like Mealy machines [2], and symbolic automata [3] [4].

However, active learning in stochastic and black-box settings has received less attention, except for Markov Decision Processes (MDPs) [5] and Stochastic Reactive Systems (SMMs) [6], where the learned probabilities converge to the true values in the large sample limit. Instead of finite automata with finite alphabets, the learning algorithms for real-time systems are complicated since continuous-time semantics. In [7], the learner can generate equivalent models with the One-clock Timed Automata (OTAs) under the real-time systems, through transforming the learning problem from the timed language observed outside to the timed language observed inside. Since the goal of Probably Approximately Correct (PAC) learning is the hypothesis generated by the learner has low error with high probability, [8] applies it to the learning of OTAs, which approximates equivalence between the learned model and the SUL by random sampling.

The contributions of this paper are as follows:

1) We present a L*-based learning algorithm for One-clock Probabilistic Timed Automata (OPTAs), where OPTAs allow for analyzing systems which operate under the existence of both probability and time.

2) In L*, the learner queries the teacher for collecting information of the SUL. Instead the existence of the teacher owning perfect knowledge of the SUL, we approximate the queries offered by the teacher under the black-box system via sampling. Besides, we apply PAC learning to check the correctness of the hypothesis generated by the learner. Thus, we carefully design the learning mechanism, the queries, the behaviors of the learner and the teacher.

3) We implemented and evaluated in the sender of CSMA/CD protocol and random examples. Experiment results show the effectiveness of our algorithm.

Structure: In Sect. II, we introduce preliminaries like OPTAs and the L* algorithm. We present L*-based learning algorithm for OPTAs in detail in Sect. III. Sect. IV discusses the evaluation and we provide a concluding in Sect. V.

II. PRELIMINARIES

Let \( \mathbb{R}_{\geq 0} \) and \( \mathbb{N} \) denote the sets of non-negative real numbers and natural numbers, respectively, and \( \mathbb{B} = \{T, F\} \) the Boolean set where T represents true. For a finite set \( Z \), a probabilistic distribution is defined by function \( \eta : Z \rightarrow [0,1] \) with \( \sum_{z \in Z} \eta(z) = 1 \), and we refer with \( \text{Dist}(Z) \) to the set of distributions over \( Z \). The concatenation of two sequences \( a, a' \in Z^* \) is denoted by \( a \cdot a' \), where \( a \) is a prefix of \( a \cdot a' \), denoted \( a \ll a \cdot a' \), and \( \text{prefixes}(a) = \{a' | a' \in Z^*: a' \ll a \} \) is the set of all prefixes of sequence \( a \). A set of sequences \( A \subseteq Z^* \) is prefix-closed, iff \( \forall a \in A : \text{prefixes}(a) \subseteq A \). Suffixes and suffix-closedness are defined analogously. We use \( \mathcal{S}(s) \in \mathbb{N} \) to denote the multiplicity of \( s \) in multiset \( \mathcal{S} \).

Probabilistic timed automata (PTAs) [9] use clocks to model real-time behaviour like classical timed automata [10]. We consider models with only one clock denoted by \( x \) in this paper. A clock valuation is a value \( \nu \in \mathbb{R}_{\geq 0} \) interpreted as the current value of clock \( x \). Let \( [\nu]_{\geq 0} \) denote that clock \( x \) is reset to 0, and \( [\nu]_{=\nu} \). The set of clock constraints over clock \( x \), denoted by \( \Phi \), is defined by the form \( \phi ::= T \land x \oplus n \land \phi \land \phi \), where \( n \in \mathbb{N} \) and \( \oplus \in \{=, <, >, \leq, \geq\} \). We use \( \nu \models \phi \) to denote a clock valuation \( \nu \) satisfies a clock constraint \( \phi \).
Definition 1. A one-clock probabilistic timed automata (OPTA) is a tuple $\mathcal{P} = (\Sigma^I, \Sigma^O, Q, q_0, x, \text{prob}, \mathcal{L})$ where

- $\Sigma^I$ and $\Sigma^O$ are finite sets of input and output symbols respectively,
- $Q$ is a set of locations, $q_0 \in Q$ is the initial location,
- $x$ is the unique clock,
- prob is the probabilistic transition relation consisting of elements with the form $(g, q, i, \eta)$, where $q \in Q$, guard $g \in \Phi$, $i \in \Sigma^I$, and $\eta \in \text{Dist}(\mathbb{B} \times Q)$, and
- $\mathcal{L} : Q \to \Sigma^O$ is a labeling function.

A state of OPTA $\mathcal{P}$ is a pair $(q, \nu)$ which transitions to either $(q, \nu + t)$ after elapsing a certain amount of time $t \in \mathbb{R}_{\geq 0}$, or $(q', [\nu])$ with probability $\eta(b, q')$ after traversing an enabled probabilistic transition $(g, q, i, \eta) \in \text{prob}$, i.e. $\nu \models g$, where outcome $(b, q') \in \mathbb{B} \times Q$ consists of a reset indicator $b$ signaling whether to reset the clock $x$ and a successor location $q'$. We use $(q, \nu) \xrightarrow{(i\cdot)\rho}[t] (q', [\nu + t])$ to denote the successor relation after both transitions, where $\mathcal{L}(q') = o$.

Given a path $\rho = (q_0, q_{n_0}) \xrightarrow{1} (q_1, q_{n_1}) \cdots \xrightarrow{t_{n_m}} (q_n, q_{n})$, we have a delay-time trace $\mathcal{L}(\rho)$, also called as delay timed trace, where $\mathcal{L}(q_k) = o_k$. Obviously, reset-delayed timed traces integrate reset information, denoted $\omega_r = o_0(i_1, t_1)(b_1, o_1) \cdots (i_n, t_n)(b_n, o_n)$ where $b_k \in \mathbb{B}$ taken from $\eta_k(b_k, q_k)$. We extend the representation of successor relation to delay timed traces $\omega$ by $(q, \nu) \xrightarrow{(i\cdot)\rho} (q', \nu')$, and $(q, \nu) \xrightarrow{(i\cdot)\rho} (q', \nu')$ if $\exists (q'', \nu'') : (q, \nu) \xrightarrow{(i\cdot)\rho} (q'', \nu'') \land (q'', \nu'') \xrightarrow{(i\cdot)\rho} (q', \nu')$.

Delay-timed traces are observed outside according to the global clock. We introduce logical timed traces and reset-logical timed traces to denote the observations inside, i.e. from the view of the local clock. We use $\Gamma$ to denote the mapping of observations outside and inside, e.g. $\Gamma(\omega_r) = \gamma_r$ represents reset logical timed trace $\gamma_r$ is the same sequence with reset delay timed trace $\omega_r$, except for the time recorded is the clock valuation $\mu_r$ after delay time $t_k$, i.e. $\gamma_r = \Gamma(\omega_r) = o_0(i_1, \mu_1)(b_1, o_1) \cdots (i_n, \mu_n)(b_n, o_n)$ where $\mu_k = t_k$ if $k = 1 \lor b_{k-1} = 1$ and $\mu_k = \mu_{k-1} + t_k$. We use $\Pi(\gamma_r) = \gamma$ to denote the projection of reset-logical timed trace $\gamma_r$ without reset information, and let $last(\gamma_r) = o_n, \nu(\gamma_r) = [\mu_n\cdot b_n]$. Let $\Sigma^I = \Sigma^I \times \mathbb{R}_{\geq 0}$ and $\Sigma^O = \mathbb{B} \times \Sigma^O$. Then $\mathcal{L}_{\forall} = \Sigma^O \times (\Sigma^I \times \Sigma^O)^*$ denotes the set of delay timed traces, and $\mathcal{L}_{\forall} = \Sigma^O \times (\Sigma^I \times \Sigma^O)^*$ denotes the set of reset-logical timed traces. We say OPTA $\mathcal{P}$ is deterministic if every delay timed trace in $\mathcal{L}_{\forall}$ corresponds to at most one path. And, OPTA $\mathcal{P}$ is complete if for all $q \in Q$ and $i \in \Sigma^I$, the guards form a partition of $\mathbb{R}_{\geq 0}$. We assume that the OPTAs we work are complete and deterministic in the following sections.

The $L^*$ algorithm [1] learns an unknown regular language $L$, which has two different roles for learning: a learner, and a teacher which is able to answer membership and equivalence queries. First the learner checks which strings are in $L$ using membership queries. While has collected sufficient information, the learner builds a hypothesis (DFA) $H$ using the membership query results. It then checks the equivalence between language $L$ and the language accepted by $H$ via equivalence query. If not, the teacher returns a counterexample to the learner for refining $H$. After processing a counterexample, the learner starts a new round until the equivalence query returns yes, i.e. $H$ accepts language $L$.

### III. LEARNING OPTAS VIA SAMPLING

Here we present a learning algorithm for OPTAs referring to the $L^*$-based algorithms of OTAs [7] and MDPs [5]. We first describe how we capture the behaviour of an OPTA.

For a timed language $L \subseteq (\Sigma \times \mathbb{R}_{\geq 0})^*$ accepted by a timed automata over alphabet $\Sigma$, we associate to a characteristic function $f : (\Sigma \times \mathbb{R}_{\geq 0})^* \to \{T, F\}$. Similarly, we explain OPTA $\mathcal{P}$ as a function $P : (\Sigma^O \times \Sigma^I)^* \to \text{Dist}(\Sigma^O) \cup \{\bot\}$.

Definition 2. For an OPTA $\mathcal{P} = (\Sigma^I, \Sigma^O, Q, q_0, x, \text{prob}, \mathcal{L})$, its characteristic function is $P$, defined for delay timed input $(i, t) \in \Sigma^I$, and delay timed trace $\omega \in \mathcal{L}_{\forall}$ as follows:

- $P(\epsilon)(\mathcal{L}(q_0)) = 1$;
- $P(\rho \cdot (i, t)) = \bot$ if $\mathcal{L}(q, \nu) : (q_0, \nu_0) \xrightarrow{(i\cdot)\rho} (q, \nu)$;
- $P(\rho \cdot (i, t))(o) = P$ otherwise if $\eta(b, q') = p > 0 \land \mathcal{L}(q') = o$, where $q, g, i, \eta \in \text{prob}$ with $\nu + t \models g$.

We say OPTAs $\mathcal{P}$ and $\mathcal{P}'$ with characteristic function $P$ and $P'$ are equivalent, iff $P = P'$.

We define reset-logical timed characteristic function $P_{lr}$, which is the characteristic function observed inside, by $P_{lr}(r, (i, \mu))(b, o) = P(\nu \cdot (i, t))(o)$ if $\gamma_r$ and $\omega$ refer to the same path, where $b \in \mathbb{B}$ is taken from $\eta(b, q')$ and $\mu = \nu(\gamma_r) + t$. We transform the learning problem to that of learning a reset-logical timed characteristic function as noted in [7], that is, we can build a hypothesis that is equivalent with the target OPTA over their reset-logical timed characteristic functions.

Theorem 1. Given OPTAs $\mathcal{P}$ and $\mathcal{P}'$ with reset-logical timed characteristic functions $P_{lr}$ and $P'_{lr}$, if $P_{lr} = P'_{lr} \Rightarrow P = P'$.

A. PAC learning of OPTAs

We assume $\mathcal{P} = (\Sigma^I, \Sigma^O, Q, q_0, x, \text{prob}, \mathcal{L})$ is the OPTA underlying the SUL, and $P$ is its characteristic function. For a hypothesis $H$ with characteristic function $H$, let $P \oplus H = \{\omega \cdot (i, t) | \omega \in \mathcal{L}_{\forall}, (i, t) \in \Sigma^I : P(\omega \cdot (i, t)) \not= H(\omega \cdot (i, t))\}$ be the symmetric difference between $P$ and $H$. We use $\mathcal{D}$ to denote a probabilistic distribution over $(\Sigma^O \times \Sigma^I)^*$. The quality of hypothesis $H$ for $\mathcal{P}$ is defined by $\mathcal{D}(P \oplus H)$.

Let $\epsilon$ and $\delta$ be the error and confidence parameters respectively. We say a learning algorithm for OPTA $\mathcal{P}$ is PAC($\epsilon, \delta$)-correct if its output learned model $H$ satisfies:

$$Pr(\mathcal{D}(P \oplus H) \leq \epsilon) \geq 1 - \delta \quad (1)$$

where $Pr$ is the probability of the event $\mathcal{D}(P \oplus H) \leq \epsilon$.

Thus, in PAC learning framework, equivalence queries require to sample delay timed test sequences, i.e. elements in $(\Sigma^O \times \Sigma^I)^*$, then test whether they belong to the symmetric difference. Theorem 2 defines the minimum number of samples required to ensure Equation 1. We discuss equivalence queries in more detail in Sect. III-D4.
Theorem 2. A learning algorithm PAC-learns an OPTA if the k-th equivalence query tests
\[ r_k = \frac{1}{\epsilon} \left( \ln \frac{1}{\delta} + \ln 2(k+1) \right). \]
random delay-timed test sequences from a fixed distribution over \((\Sigma^O \times \Sigma^I)^*\) without finding a counterexample.

B. Queries

Let \( \lambda = \gamma \cdot e \in (\Sigma^O \times \Sigma^I)^* \) be a logical-timed test sequence, where \( \gamma \) is a logical-timed trace, and \( e \) is called as logical-timed continuation sequence. Obviously, reset-logical-timed test sequences is the logical-timed test sequences extending with reset information.

The teacher gains information about the reset-logical-timed characteristic function by sampling system traces. Let \( S \) be the multisets of reset-logical-timed traces collected from the SUL during learning. Thus, the teacher offers four queries:

Frequency queries: take a logical-timed test sequence \( \lambda \) as input, and return the corresponding reset-logical-timed test sequence \( \pi(\lambda) \) and output frequencies \( \text{freq}(\lambda) : \Sigma^O \rightarrow \mathbb{N} \) collected in multiset \( S \).

Complete queries: given a logical-timed test sequence \( \lambda \), complete query \( \text{cq}(\lambda) \) returns whether \( \text{freq}(\lambda) \) has sufficient information to approximate the true distribution.

Refine queries: given a set rare of logical-timed test sequences which require to refine the knowledge about the SUL, refine query \( \text{rq}(\text{rare}) \) returns a multisets of reset-logical-timed traces sampled from the SUL along rare.

Equivalence queries: given a hypothesis \( H \), and the parameters: error \( \epsilon \), confidence \( \delta \) and the amount of previous equivalence queries \( k \), equivalence query \( \text{eq}(H, \epsilon, \delta, k) \) returns a pair \((r, ctx)\) where \( r \) is the Boolean value signaling whether \( H \) passes all tests, and \( ctx \) is a counterexample.

C. Learner

1) Timed observation table: Timed Observation table is used to store information growing during learning in our work.

Definition 3. A timed observation table is a tuple \( T = (\Sigma^I, S, R, E, f, \psi) \), where \( \Sigma^I = \Sigma^I \times \mathbb{R}_{\geq 0} \) is a infinite set of logical-timed inputs, \( S, R \subset T \cdot R \) are finite sets of reset-logical-timed traces, \( E \subset (\Sigma^I \times \Sigma^O)^* \times \Sigma^I \) is a suffix-closed set of logical-timed continuation sequences, \( \psi : S \cup R \rightarrow 2^{\Sigma^I} \), and \( \forall r, r' \in S \cup R, e \in E \cup \psi(\gamma) : f(\gamma \cdot e) = \text{freq}(\Pi(\gamma) \cdot e) \).

Specifically,

- \( S \) is called the set of prefixes, \( R \) is called the boundary, and \( E \) is called the set of suffixes.
- \( S \) and \( R \) are disjoint, and \( S \cup R \) is prefix-closed.
- \( \forall \gamma \in S \cup R, i \in \psi(\gamma), o \in \Sigma^O : \gamma \cdot i \cdot o \in R \) if \( \text{freq}(\Pi(\gamma) \cdot i)(o) > 0 \).

We represent the content of a row by \( \text{row}(\gamma_i)(e) = f(\gamma_i \cdot e) \) for \( \gamma_i \in S \cup R, e \in E \cup \psi(\gamma) \). We initially set \( S = \{ L(q_0) \} \), where \( L(q_0) \) is the initial output of the SUL, \( E = \{(i, 0)|\Sigma^I\} \), and \( \psi(L(q_0)) = \{(i, 0)|\Sigma^I\} \).

Since can not directly determine equivalence of rows, we apply Hoeffding bounds [11] to determine whether two logical-timed test sequences produce statistically different distributions. Put differently, for logical-timed test sequences \( \lambda_1, \lambda_2 \), we approximate \( P_{\lambda_1}(\pi(\lambda_1)) \neq P_{\lambda_2}(\pi(\lambda_2)) \) by checking whether \( \text{freq}(\lambda_1) \) and \( \text{freq}(\lambda_2) \) have been sampled from different distributions. Thus, we say \( \lambda_1, \lambda_2 \) produce statistically different distributions, denoted \( \text{diff}(\lambda_1, \lambda_2) \), iff one of the following conditions holds:

1) \( \text{cq}(\lambda_1) \land \text{cq}(\lambda_2) \land (n_1 > 0 \land n_2 > 0) \), or
2) \( n_1 > 0 \land n_2 > 0 \land \exists \alpha \in \Sigma^O : \left| \frac{f_1(o)}{n_1} - \frac{f_2(o)}{n_2} \right| > \frac{1}{\sqrt{n_1}} + \frac{1}{\sqrt{n_2}} \sqrt{\frac{1}{2} \ln \frac{2}{\alpha}} \) (3)

where \( f_k = \text{freq}(\lambda_k), n_k = \sum_{o \in \Sigma^O} f_k(o) \) for \( k = 1, 2 \), and \( \alpha \) specifies the confidence level \((1-\alpha)^2\).

We say \( f(\gamma_i \cdot e) \) and \( f(\gamma_i' \cdot e') \) are compatible, denoted \( f(\gamma_i \cdot e) \approx f(\gamma_i' \cdot e') \) if \( \text{diff}(\Pi(\gamma_i) \cdot e, \Pi(\gamma_i') \cdot e') \) for \( \gamma_i \in S \cup R, e \in E \cup \psi(\gamma) \). We extend this notion to rows, i.e. we say two rows labeled \( \gamma_r \) and \( \gamma_r' \) are compatible, denoted \( \text{row}(\gamma_r) \approx \text{row}(\gamma_r') \), if all of the following conditions hold: (1) \( \text{last}(\gamma_r) = \text{last}(\gamma_r') \), (2) \( \forall e \in E : \text{row}(\gamma_r(e)) \approx \text{row}(\gamma_r')(e) \), and (3) \( \forall e \in \psi(\gamma_r) \cap \psi(\gamma_r') : \text{row}(\gamma_r)(e) \approx \text{row}(\gamma_r')(e) \).

The hypothesis generated from a timed observation table \( T \) is well-formed if \( T \) is prepared, which satisfies all of the following properties:

- reduced: \( \forall s, s' \in S \Rightarrow \text{row}(s) \neq \text{row}(s') \)
- closed: \( \forall r \in R, \exists s \in S : \text{row}(s) \approx \text{row}(r) \)
- consistent: for all compatible pairs of rows labeled \( \gamma_r, \gamma_r' \in S \cup R \), the logical-timed input \( i \in \psi(\gamma_r) \cap \psi(\gamma_r') \), and output \( o \in \Sigma^O \), we have \( \text{row}(\gamma_r \cdot i : (\gamma_r, o)) \approx \text{row}(\gamma_r' \cdot i : (\gamma_r', o)) \) if \( \gamma_r \cdot i : (\gamma_r, o) \neq \text{row}(\gamma_r' \cdot i : (\gamma_r', o)) \) in \( S \cup R \).
- evidence-closed: \( \forall s \cdot e \in E : \text{row}(\gamma_e) \approx \text{row}(\gamma_e)(e) \) is the longest prefix such that \( \gamma_e \in S \).

We apply repeatedly following operations to get a prepared table \( T \). We move \( r \in R \) to \( S \) and update \( \psi(r) = \psi(r) \cup \{(i, 0)|\Sigma^I\} \) as noted in [7], if \( r \) breaks the closedness. Notably, we add column \( i \in \psi(r) \) to \( E \) if \( \text{row}(r)(i) \) makes \( \text{row}(r) \) different from all of the rows in \( S \). As for consistent, if there exist \( \gamma_r, \gamma_r' \in S \cup R \) such that \( \text{row}(\gamma_r) \approx \text{row}(\gamma_r') \) but \( \text{row}(\gamma_r \cdot i : (\gamma_r, o)) \neq \text{row}(\gamma_r' \cdot i : (\gamma_r', o)) \), we add suffix \( i \cdot o \cdot e \) to \( E \), where \( e \in E \) such that \( f(\gamma_r \cdot i : (\gamma_r, o) \cdot e) \neq f(\gamma_r' \cdot i : (\gamma_r', o) \cdot e) \). To make table \( T \) evidence-closed, we add all the prefixes \( \gamma_r \cdot i \in \text{prefixes}(\pi(\Pi(s) \cdot e)) \) such that \( \gamma_r \in S \) to the table \( T \), i.e. add \( \gamma_r \) to \( R \) and update \( \psi(\gamma_r) = \psi(\gamma_r) \cup \{i\} \).

However, compatibility is not transitive in general, that is, a row in \( R \) may be compatible with multiple rows in \( S \). Thus we create compatible classes for partitioning \( R \) as noted in [5], where every trace \( \gamma_r \in S \cup R \) in compatible class \( cq(s) \) are compatible with its representative \( s \in S \). We use \( \text{rep}(\gamma_r) = s \) to denote \( s \) is the representative of trace \( \gamma_r \), which has the largest rank among \( S \), where rank is before-defined.

2) Hypothesis generation: We build a hypothesis \( H \) from a prepared timed observation table \( T \), where table \( T \) is used to construct an intermediary automaton \( M \), then \( M \) is transformed to hypothesis \( H \) via partition function [7].
We define an intermediary automata is a tuple \( M = (Q_M, \Sigma_M, \Sigma^I_M, \Sigma^O_M, \text{prob}_M, \mathcal{L}_M) \), where \( Q_M \) is a finite set of states, \( \Sigma_M \subset \Sigma^I \) is a finite set of logical-timed inputs, \( \Sigma^O_M \) is a finite set of outputs, \( q_0^M \in Q_M \) is the initial state, \( \text{prob}_M \subseteq Q_M \times \Sigma_M \times \text{Dist}(\mathbb{B} \times Q_M) \) is the transition relations, and \( \mathcal{L}_M : Q_M \rightarrow \Sigma^O_M \) is a labelling function.

We build an intermediary automata \( M = (Q_M, \Sigma_M, \Sigma^O_M \cup \{\text{undef}\}, q_0^M, \text{prob}_M, \mathcal{L}_M) \) from a prepared timed observation table \( T = (\Sigma^I, S, R, E, f, \psi) \), as follows:

- \( Q_M = \{\text{last}(s), \text{row}(s)\} \subset S \cup \{q_{ud}\} \)
- for \( q = (o, \text{row}(s)) \in Q_M \), \( \mathcal{L}_M(q) = o \) for \( q_{ud} \in Q_M \), set \( \mathcal{L}_M(q_{ud}) = \text{undef} \)

- \( q_0^M = (\text{L}(q_0), \text{row}(\text{L}(q_0))) \)
- \( \Sigma^I_M = \{i \in \psi(\gamma_r) | \gamma_r \in S \cup R\} \)
- for \( \gamma_r \in S \cup R, (i, \mu) \in \psi(\gamma_r) \): let \( s = \text{rep}(\gamma_r), q = \langle \text{last}(s), \text{row}(s) \rangle, n = \sum_{(i, \mu) \in \psi(\gamma_r)} \text{freq}(\Pi(\gamma_r) \cdot (i, \mu))(o) \), and \( \text{rep}(q, (i, \mu)) = \lambda \) where \( \lambda \in \Pi(\gamma_r) \cdot (i, \mu) \). Let \( \gamma_r' \in c(g)(s, (i, \mu)) \in \psi(\gamma_r) \). If \( \text{freq}(\Pi(\gamma_r), (i, \mu)) \) is not equal to \( \text{freq}(\Pi(\gamma_r'), (i, \mu)) \) with the largest rank,
  1) if \( \text{eq}((\gamma_r), (i, \mu)) \) and \( \#(\gamma_r') < c(g)(s, (i, \mu)) \) : \( \text{prob}_M = \text{prob}_M \cup \{q', (i, \mu), \eta_{ud}\} \), where \( \eta_{ud}(T, q_{ud}) = 1 \)
  2) otherwise \( \text{eq}((\gamma_r), (i, \mu)) \) and \( n > 0, \text{prob}_M = \text{prob}_M \cup \{q, (i, \mu), \eta_{ud}\} \)

where \( \eta_{ud}(T, q_{ud}) = 1 \) if \( q_{ud} \) is reachable.

We create a state \( q \) for every \( s \in S \), and transitions from state \( q \) lead to the representatives of the extensions \( \gamma_r \cdot (i, \mu) \) for each \( \gamma_r \in \psi(\gamma_r) \). The representative frequencies \( \text{rep}(q, (i, \mu)) \) are estimated by the representative frequencies \( \text{rep}(q, (i, \mu)) \).

The representatives are used to refer to the compatible frequencies to the same distributions. If we have not sufficient information for a logical-timed input, we create a transition to a sink state \( q_{ud} \).

We receive a OPTA \( \mathcal{H} = (\Sigma^I, \Sigma^O, Q_H, q_0^H, x, \text{prob}_H, \mathcal{L}_H) \) from an intermediary automata \( M = (Q_M, \Sigma_M, \Sigma^O \cup \{\text{undef}\}, q_0^M, \text{prob}_M, \mathcal{L}_M) \) as noted in [7], where \( Q_H, q_0^H \) and \( \mathcal{L}_H \) is the same with \( M \), and \( \text{prob}_H \) are transformed from \( \text{prob}_M \) one by one: for a group of probabilistic transitions with the same source \( q \in Q_H \) and input \( i \in \Sigma^I \), their guards are a partition of \( \mathbb{R}^n \) such that each guard \( q \) includes the clock valuation \( \mu \) recorded in the corresponding transition of \( M \), and we denote as \( I_{q,i}(\mu) = \mu \) with \( \mu' = g \).

3) Learning algorithm: Algorithm 1 implements the learning algorithm for OPTAs via sampling. First, it initializes a timed observation table \( T = (S, R, E, f, \psi) \) with \( S = \{\mathcal{L}(q_0)\} \) where \( \mathcal{L}(q_0) \) is the initial output of SUL, \( E = \{(i, 0) | i \in \Sigma^I\} \), and \( \psi(\mathcal{L}(q_0)) = \{(i, 0) | i \in \Sigma^I\} \). Then, it performs the main loop until \( \text{equivalent} = \text{yes} \) which represents the final equivalence query terminates without finding a counterexample. The loop starts with a refine query \( \text{rq} \) with the set of incomplete logical-timed test sequences of table \( T \), i.e. \( \text{get_incomplete_seq}(T) = \{\Pi(\gamma_r) \cdot e | \gamma_r \in S \cup R, e \in E \cup \psi(\gamma_r) : -\text{eq}(\Pi(\gamma_r) \cdot e)\} \), then updates the extensions, where \( \text{Lt}(S \cup R) = \{\gamma_r \cdot \text{io} | \gamma_r \in S \cup R, i \in \psi(\gamma_r), o \in \Sigma^O \}, \text{freq}(\Pi(\gamma_r) \cdot i)(o) > 0\}, \) and fills table based on multiset \( S \). We adjust table \( T \) to be prepared before hypothesis generation. After building a hypothesis \( H \), we determine whether to perform equivalence queries depending on the uncertainty of table \( T \), which is discussed in detail later. We have \( \text{equivalent} = \text{no} \) if not. Otherwise if finding a counterexample \( \text{ce} \), \( \text{ce} \) with normalizing [7] and adding all the prefixes of \( \text{ce} \) to table \( T \), i.e. \( \forall \gamma_r \cdot i \in \text{prefixes}(\text{ce}) : R = R \cup \{\gamma_r \} \wedge \psi(\gamma_r) = \psi(\gamma_r) \cup \{i\} \). Once the loop stops, we output the hypothesis.

Uncertainties in time observation tables mainly arise from the compatibility checks. Put differently, for \( r \in R \), the state reached by \( r \) may be ambiguous if \( \text{row}(r) \) is compatible with multiple rows \( s \in S \). We quantify the uncertainty as the ratio \( r_{\text{uamb}} \) of ambiguous rows and all of the rows, where \( r_{\text{uamb}} \) is ambiguous rows are compatible with a row in \( S \). Besides, there exist unknown distributions in hypothesis \( H \) if location \( q_{ud} \) is reachable. Thus, to decrease the amount of unnecessary equivalence queries, we perform equivalence queries once location \( q_{ud} \) is unreachable and \( r_{\text{uamb}} \geq m_{\text{uamb}} \).

Algorithm 1 Learning of OPTAs by sampling

Input: timed observation table \( T = (\Sigma^I, S, R, E, f, \psi) \),

decider of answering \( \text{freq}, \pi, \text{eq}, \text{rq} \) and \( \text{eq} \)

Output: final hypothesis \( H \)

1: \( T \leftarrow \text{initialized}(T) \)
2: \( k \leftarrow 0 \)
3: \text{repeat}
4: \( k \leftarrow k + 1 \)
5: \( \text{rare} \leftarrow \text{get_incomplete_seq}(T) \)
6: \( S \leftarrow S \cup \text{rq} \leftarrow \text{eq}(\text{rare}) \)
7: \( R \leftarrow R \cup \text{Lt}(S \cup R) \)
8: \text{for all} \( \gamma_r \in S \cup R, e \in E \) do
9: \( \text{f}(\gamma_r \cdot e) \leftarrow \text{freq}(\Pi(\gamma_r) \cdot e) \)
10: \text{while} \( T \) is not prepared do
11: \( T \leftarrow \text{make_prepared}(T) \)
12: \( H \leftarrow \text{build_hypothesis}(T) \)
13: \text{if} \( \text{enable_eq}(T, H) \) then
14: \( \text{equivalent}, \text{ce} \leftarrow \text{eq}(H, k, \epsilon, \delta) \)
15: \text{if} \( \text{ce} \neq \text{none} \) then
16: \( T \leftarrow \text{process_ce}(\text{ce}, T) \)
17: \text{else}
18: \( \text{equivalent} \leftarrow \text{no} \)
19: \text{until} \( \text{equivalent} = \text{yes} \)
20: \text{return} \( H \)

D. Teacher

We describe how to answer the four queries provided by the teacher in the following. We first introduce the SUL operations providing for the teacher, which include: (1) \( \text{reset} \): reset the SUL and return \( \mathcal{L}(q_0) \); (2) \( \text{step} \): perform a delay-timed input, change the current state according to the distribution of the
enabled transition, and return \((b, L(q'))\), where \((b, q')\) is the selected outcome. Here we assume that the SUL is smart to answer the reset information in transitions.

1) Frequency queries: For a logical-timed test sequence \(\lambda\), let \(\lambda = \gamma \cdot e\) where \(\gamma\) is the longest logical-timed trace with \(\exists r_\gamma \in S: \Pi(\gamma r_\gamma) = \gamma\). The frequency query for \(\lambda\) returns pair \((\pi(\lambda), \text{freq}(\lambda))\), where \(\pi(\lambda) = \gamma \cdot r_\gamma \cdot e\), in which \(e\) sets all of the reset indicators of \(e\) to \(T\), and \(\text{freq}(\lambda)(o) = S(\gamma \cdot r_\gamma \cdot e \cdot o)\) for all \(o \in \Sigma^0\).

2) Complete queries: We have \(S(\gamma r_\gamma) \leq S(\gamma r_\gamma')\) for all \(\gamma r_\gamma' \in \text{prefixes}(\gamma r_\gamma)\), since we add all of the prefixes of samples to \(S\). For a logical-timed test sequence \(\lambda\), we assume that \(c(q(\lambda)) = T\) if \(\sum_{o \in \Sigma^0} S(\pi(\lambda) \cdot o) \geq n_c\), and we have already seen all of the outputs after \(\lambda\). Thus the extensions of \(\lambda \cdot o\) are complete if \(S(\pi(\lambda) \cdot i) = 0\). Besides, for \(\gamma r_\gamma \in S\), we have the extensions of \(\Pi(\gamma r_\gamma) \cdot (i, \mu)\) are also complete if \(\mu\) less than the clock valuation of the state reached by \(\gamma r_\gamma\).

3) Refine queries: The procedure of refine queries we use is similar with [5], in which a prefix tree is the compact representations of the set \(\text{rare}\) consisting of sequences requiring to refine the knowledge, and new sampled SUL traces generated by directed random walks on the prefix tree. But the prefix tree we use is a tree with edges labeled logical-timed inputs, and nodes labeled outputs. Notably, to represent as a prefix tree, the sequences in \(\text{rare}\) are extended to logical-timed traces by add a special output \(\text{leaf} \notin \Sigma^0\) at the end of every sequence. To get the delay-time inputs used to operation step, we record the current clock valuation during sampling. We receive \(t_{\text{resample}}\) reset-logical-timed traces after performing a refine query.

4) Equivalence queries: Recall that the framework of PAC learning for OPTAs. We face the difficulty that a delay-timed test sequences \(\omega \cdot i \in (\Sigma^0 \times \Sigma^I)^*\) sampled randomly cannot directly determine \(P(\omega \cdot i) \neq H(\omega \cdot i)\). Thus, we apply two strategies to find counterexamples as in Algorithm 2. We first compute the number of samples \(r_k\). Then, we sample delay-timed traces from a distribution \(D\), and obtain the corresponding reset-delay-timed traces on the SUL by testing. The sampling mechanism is discussed in detail later. We return unreachable samples on \(H\) as counterexamples, or check for consistency between multiset \(S\) and hypothesis \(H\) respect to these samples according to Theorem 3. To not produce spurious results for \(\text{diff}\), we perform refine queries for gaining more information about samples, until all of the distributions of samples are unambiguous, where \(\text{unamb(tests)} = \frac{|\{\text{tests} \setminus \text{eq(tests)}\}|}{|\text{tests}|}\). Under the unambiguous setting, we have \(H\) is PAC\((\epsilon, \delta)\)-correct if without finding a counterexample.

**Theorem 3.** Let \(C \subseteq \{\omega \cdot (i, t) \in (\Sigma^0 \times \Sigma^I)^* | P(\omega \cdot (i, t) \neq \perp)\}\), and multiset \(S\) has \(n\) samples of \(C\). Given \(\alpha_n\) such that \(\sum \alpha_n \alpha_n < \infty\), then with probability one, \(\forall \omega \cdot (i, t) \in C: P(\omega \cdot (i, t)) \neq H(\omega \cdot (i, t)) \Rightarrow \text{diff}(\gamma \cdot (i, \mu), \text{repf}(q((i, I_q(i, \mu))))),\)
where \(\gamma = \Gamma(\omega), q \in Q_H: (q_{\gamma H}, \nu_0) \xrightarrow{\omega} (q, \nu), \text{and } \mu = \nu + t,\) except for finitely many \(n\).

The purpose of a sample mechanism is to sample the SUL traces different from the hypothesis \(H\). Without Hoeffding checking, unreachable sampled traces on \(H\) diverge the SUL and the hypothesis. The sampling distribution \(D\) is that we randomly select delay-timed input \((i, t) \in \Sigma^I\), test iteratively the SUL to observe output, and stop with probability \(p_{\text{stop}}\). Besides, we choose some traces to extend randomly. More techniques used to improving sampling efficiency, can be found in [8].

**Theorem 4.** Algorithm 1 terminates and outputs a model that has an average error less than or equal \(\epsilon\) with the the probability of at least \(1 - \delta\), except for finitely many \(n\).

IV. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In our work, we want to learn a model close to the true model, since equivalence can hardly be achieved. Our experiments aim to measure the distance between the learned models and the true models. More concretely, 2000 test cases generated from true models are calculated first whether executable on the learned models. If executable, we calculate the Kullback-Leibler Divergence (KLD) of the output distributions produced by both models, and record executable test cases are passing. And we compare the probability of the temporal properties for the true model and the learned model, if the true model can perform probabilistic model check by Prism [12]. We implemented our algorithm in JAVA, and performed the experiments on a PC with 16 GB RAM, an Intel Core i7-9750 CPU with 2.6 GHz and running Windows 10.
TABLE I: Results for learning the CSMA/CD protocol

<table>
<thead>
<tr>
<th>Case ID</th>
<th>ϵ, δ</th>
<th>r/pass</th>
<th>KLD</th>
<th># trace</th>
<th>t/mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>4_2_2_1</td>
<td>0.01</td>
<td>0.9977</td>
<td>0.2836</td>
<td>2196039</td>
<td>102.047</td>
</tr>
<tr>
<td>4_2_3_20</td>
<td>0.01</td>
<td>0.9996</td>
<td>0.3131</td>
<td>2305877</td>
<td>134.190</td>
</tr>
<tr>
<td>6_2_3_20</td>
<td>0.01</td>
<td>0.9988</td>
<td>0.1421</td>
<td>2401363</td>
<td>184.100</td>
</tr>
<tr>
<td>6_2_4_20</td>
<td>0.01</td>
<td>0.9925</td>
<td>0.1462</td>
<td>1978498</td>
<td>85.787</td>
</tr>
<tr>
<td>6_2_5_20</td>
<td>0.01</td>
<td>0.9951</td>
<td>0.1303</td>
<td>1940146</td>
<td>89.791</td>
</tr>
<tr>
<td>6_2_6_20</td>
<td>0.01</td>
<td>0.9974</td>
<td>0.1625</td>
<td>1848045</td>
<td>90.774</td>
</tr>
<tr>
<td>10_2_7_40</td>
<td>0.1</td>
<td>0.9000</td>
<td>0.0718</td>
<td>1330212</td>
<td>46.045</td>
</tr>
<tr>
<td>10_2_8_40</td>
<td>0.1</td>
<td>0.9565</td>
<td>0.0458</td>
<td>3363402</td>
<td>176.929</td>
</tr>
<tr>
<td>10_2_9_40</td>
<td>0.001</td>
<td>0.9991</td>
<td>0.0433</td>
<td>1587538</td>
<td>9277.404</td>
</tr>
</tbody>
</table>

TABLE II: Results on random examples

<table>
<thead>
<tr>
<th>Case ID</th>
<th>ϵ, δ</th>
<th>r/pass</th>
<th>KLD</th>
<th># trace</th>
<th>t/mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>4_2_2_1</td>
<td>0.01</td>
<td>0.9977</td>
<td>0.2836</td>
<td>2196039</td>
<td>102.047</td>
</tr>
<tr>
<td>4_2_3_20</td>
<td>0.01</td>
<td>0.9996</td>
<td>0.3131</td>
<td>2305877</td>
<td>134.190</td>
</tr>
<tr>
<td>6_2_3_20</td>
<td>0.01</td>
<td>0.9988</td>
<td>0.1421</td>
<td>2401363</td>
<td>184.100</td>
</tr>
<tr>
<td>6_2_4_20</td>
<td>0.01</td>
<td>0.9951</td>
<td>0.1303</td>
<td>1940146</td>
<td>89.791</td>
</tr>
<tr>
<td>6_2_5_20</td>
<td>0.01</td>
<td>0.9974</td>
<td>0.1625</td>
<td>1848045</td>
<td>90.774</td>
</tr>
<tr>
<td>6_2_6_20</td>
<td>0.01</td>
<td>0.9978</td>
<td>0.0734</td>
<td>2197287</td>
<td>120.542</td>
</tr>
<tr>
<td>6_2_7_20</td>
<td>0.01</td>
<td>0.9999</td>
<td>0.0588</td>
<td>4095301</td>
<td>493.074</td>
</tr>
<tr>
<td>10_2_7_40</td>
<td>0.1</td>
<td>0.9000</td>
<td>0.0718</td>
<td>1330212</td>
<td>46.045</td>
</tr>
<tr>
<td>10_2_8_40</td>
<td>0.1</td>
<td>0.9565</td>
<td>0.0458</td>
<td>3363402</td>
<td>176.929</td>
</tr>
<tr>
<td>10_2_9_40</td>
<td>0.001</td>
<td>0.9991</td>
<td>0.0433</td>
<td>1587538</td>
<td>9277.404</td>
</tr>
</tbody>
</table>

A. CSMA/CD Protocol

We consider to learn the sender in IEEE 802.3 CSMA/CD protocol, which can be specified and verified formally within the framework of probabilistic timed automata [13]. In our setting, the corresponding OPTA P to be learned is configured to have |Q| = 8 locations and with two WAIT states moved with probability 0.5, |Σ| = 5, and the propagation delay is 26µs, the time to send a data packet is 808µs. For the evaluation of the CSMA/CD protocol, we perform probabilistic model checking for the minimum probability of reaching the goal done within a varying number of time. We set the sampling parameters as n_sample = 100, p_stop = 0.5. Let m_unamb = 0.999 for the enable condition of equivalence queries. As the parameters of PAC, we set the error parameter ϵ = 0.001 and the confident parameter δ = 0.001. We set α = 0.001 for the compatibility check, and the complete threshold n_c = 50.

Table I shows the measurement results for learning the CSMA/CD protocol. The passing ratio achieves 100%, and the absolute difference to the true probabilities is at most 0.0004 in probabilistic model checking.

B. Random experiments

We randomly generated 90 OPTAs in nine groups, with each group labelled n_m_c_k, having different numbers of locations, size of inputs, size of outputs, and maximum constant appearing in clock constraints. We set the sampling parameters as n_sample = 200, p_stop = 0.5, m_unamb = 0.999 for the enable condition of equivalence queries, α = 0.01 for the compatibility check, and the complete threshold n_c = 50. As shown in Table II, the passing ratios at least 99%, and the sum of KLD for the passing test cases are at most 0.5 in all cases. While ϵ and δ are smaller, the passing rate increases.

1To ensure the completeness of 𝑃, we complete it as noted in [7], i.e. any unspecified behavior in the original model is regarded as transitioning to a sink location

V. CONCLUSION

In this paper, we present a learning algorithm for one-clock probabilistic timed automata based on the framework of L* via sampling. The traces sampled from the SUL are used to infer the structure, estimate transition probabilities, and check the compatibility between the target model and the hypothesis generated by learner. We apply the PAC learning framework to our learning algorithm for quantifying the learned model under the black-box setting. We evaluate our learning algorithm on randomly generated examples and the sender of CSMA/CD protocol. Future works include combines our learning algorithm with learning-based verification techniques for case studies on stochastic systems.

REFERENCES

Session MLDLA: Machine Learning and Deep Learning Algorithms
DeepMultiple: A Deep Learning Model for RFID-based Multi-object Activity Recognition

Shunwen Shen¹, Lvqing Yang¹*, Sien Chen²,³, Wensheng Dong⁴, Bo Yu⁴, Qingkai Wang⁵,⁶
¹School of Informatics, Xiamen University, Xiamen, China
²School of Navigation, Jimei University, Xiamen, China
³School of Management, Xiamen University, Xiamen, China
⁴Zijin Zhixin(Xiamen)Technology Co., Ltd., Xiamen, China
⁵State Key Laboratory of Process Automation in Mining & Metallurgy, Beijing, China
⁶Beijing Key Laboratory of Process Automation in Mining & Metallurgy, Beijing, China
*The corresponding author email: lqyang@xmu.edu.cn

Abstract—Wireless sensing techniques for Human Activity Recognition (HAR) have been widely studied in recent years. At present, the research on HAR based on Radio Frequency Identification (RFID) is changing from the tag attachment method to the tag non-attachment method. Affected by multipath, the current solutions in tag non-attachment scenarios mainly focus on single-object activity recognition, which is not suitable for multi-object scenarios. To address these issues, we propose DeepMultiple, a novel tag non-attachment activity recognition model for multi-object. The model first preprocesses the raw signal with filter and phase calibration, then it applies dilated convolution in the frequency domain to extract multi-object activity features, finally ProbSparse is used to optimize the vanilla Transformer-based Encoder to enhance the activity recognition ability. We deployed a single reader and antenna for multi-object activity tracking during the experiments to reduce deployment difficulties. Extensive experimental results show that DeepMultiple can recognize ten types of multi-object activities with 98.12% precision under different challenging settings, which has excellent performance compared with several state-of-the-art methods.

Index Terms—RFID, Multi-object, Human Activity Recognition, Deep Learning

I. INTRODUCTION

With the development of deep learning technology, human activity recognition (HAR) has become one of the most important tasks in ubiquitous computing. It has attracted widespread attention from industry and academia. HAR based on cameras and portable wearable devices has problems such as high line-of-sight (LoS) requirements, unfriendly privacy protection (e.g. cameras), and real-time body attachment (e.g. wearable devices). RFID technology has become a new choice in the field of HAR as its low cost, small form size, and convenient deployment [1]. It has been widely used in various scenarios such as patient health monitoring [2], motion guidance in gyms [3], and activity monitoring on market shelves [4].

There are two tag placement methods for RFID-based HAR. (1) tag-attached method, tag-attached method employs RFID tags to attach to the human body (i.e. reference [5], reference [6]). (2) tag non-attachment method, tag non-attachment method apply multiple tags to fix in the environment as fixed references (i.e. reference [7]). The tag attachment method needs to attach RFID tags to specific objects or users to track their movements and infer their activities. This method has the inconvenience of equipment and privacy violations. Therefore, more and more researchers turn their attention to tag non-attachment method. But there are still the following challenges in the tag non-attachment scenario. First, it is easy to cause signal attenuation on the direct propagation link between the tag and the antenna due to environmental interference. So more antennas need to be brought in to reduce the multipath interference caused by the environment. Second, current research on RFID-based activity recognition mainly focuses on simple scenarios, that is, a person in an open environment. Since the interaction between multiple objects is more abundant, the interaction signals after backscattering are inevitably mixed. It will be more difficult to extend to multi-object activity recognition.

To address the above challenges, we propose DeepMultiple, a novel model for multi-object activity recognition in complex multipath environments. We use both RSSI and phase as input for getting more useful information from them. The model splits the preprocessed time-series data by sliding time windows and uses dilated convolution and selective attention mechanism to extract available features in complex multipath information to realize multi-object activity recognition. Besides, we deployed only a single antenna for activity identification during the experiment.

The contributions of this paper are summarized as follows.

• We propose a challenging multi-object activity recognition scenario without tag attachment. Compared to several state-of-the-art methods, experimental results demonstrate the superiority of our proposed model.

• To the best of the authors’ knowledge, the DeepMultiple is the first model that applies dilated convolution in the RFID field to merge the spatial feature from different objects, and its validity is proven by ablation experiments.

• DeepMultiple optimizes the vanilla Transformer-based Encoder by ProbSparse attention mechanism, reduces the calculation of the parameters, and realizes multi-object HAR with only 2600 training samples.

DOI reference number: 10.18293/SIEK2023-138
II. RELATED WORK

Tag-attached: Currently, there is a lot of work dedicated to human activity recognition based on RFID technology. Reference [10] attaches RFID tags to the user’s back and recognizes the user’s habitual sitting posture by establishing the correlation between the phase change of the tags and the sitting postures. TagBreathe [6] attaches RFID tags to users and measures the tiny movement of the human chest to monitor respiration. By attaching passive RFID tags on the dumbbells and leveraging the Doppler shift profile of the reflected backscatter signals, FEMO [5] provides an integrated free-weight exercise monitoring system. RF-idraw [11] uses antenna array and beam steering technology to track the trajectory of the marked object. Tagoram [12] proposes a Differential Augmented Hologram (DAH) which will facilitate the instant tracking of the mobile RFID tag with millimeter accuracy. The above research is carried out around the tag attached to the detector, however, many activities do not directly interact with RFID-tagged objects. Empirical results have shown that RFID signals can be influenced by nearby human activities even if the objects are not moved [8].

Tag non-attachment: In recent years, more and more researchers have studied activity recognition in tag non-attachment scenarios. RFIPad [7] enables in-air handwriting without tags attached. RF-Care [13] aims to use passive RFID arrays to establish a tag non-attachment activity recognition for older people. Tagfree [8] is a pioneering work that aims to distinguish and use useful features contained in complex multipath information. Although the above research is based on tag non-attachment, more devices need to be used for accurate identification, which increases deployment costs. And these methods do not have the ability of multi-object recognition.

III. PRELIMINARIES

In this section, we will introduce the basic theory and data preprocessing methods in RFID.

A. RFID Communication Mechanism

Received Signal Strength Indicator (RSSI). The wireless signal will be divided into multiple signals during the propagation process. After the reader sends signals to the surrounding environment, multiple signal signals are superimposed and returned to the reader. The signal strength received by the reader can be modeled as:

\[ S = \sum_{i=1}^{N} h \cdot |F_i| e^{j\theta} \]  

(1)

where \( h \) represents the attenuation coefficient in the signal propagation process, \( F_i \) is the amplitude of the \( i \)-th communication path signal, and \( \theta \) is the phase of RFID signal. Due to the influence of multi-object interaction and multipath effects in the environment, the signal \( S \) consists of three parts: the \( S_{tag} \) returned directly by the tag, the \( S_{env} \) returned by the environment and the \( S_{objects} \) returned by the multi-object interaction as shown in Fig.3.

\[ S = S_{tag} + S_{env} + S_{objects} \]  

(2)

From the above two formulas, we can conclude that in a complex multipath environment, the signal received by the reader is mixed with multiple signals, and it will be affected by distance and attenuation coefficient.

Phase. For a single-reader RFID propagation environment, the phase information of the signal received from the \( k \)-th tag provided by the reader is \( \theta = \text{mod}(\theta_p + \theta_a + \theta_r + \theta_t, 2\pi) \), \( \theta_p = 2\pi d/\lambda \). \( d \) is the propagation distance of RFID signal, \( \lambda \) represents the wavelength of the RFID signal. \( \theta_a, \theta_r, \theta_t \) represent the phase jump caused by the antenna transmission circuit, receiver circuit and tag reflection characteristics.

B. Data Preprocessing

1) Phase Calibration: As shown in Fig.1(a) and Fig.1(b), we find that RSSI is less sensitive to human activity, but still detects fluctuations caused by activity, while the phase is very sensitive to information, but not accurate. The main reason for the inaccurate raw phase is caused by the frequency hopping mechanism. Previous research has shown that frequency hopping can lead to significant phase shifts due to the phase difference of the oscillators and the non-uniform frequency response of the antenna [19]. And according to PRELIMINARIES, we know that the phase shift will also be affected by transmission, receive, and reflection links.

We use phase unwrapping [15] and phase smoothing to eliminate this phase shift. The smoothing algorithm works by collecting an initial phase measurement, which takes about 10 seconds for a stationary tag. It can be written as:

\[ \theta(t) = \theta_j(t) - \bar{\theta}_j + \bar{\theta}_d \]  

(3)

where \( \theta_j(t) \) denote the measured phase at frequency \( f_j \) at time \( t \). \( \bar{\theta}_j \) and \( \bar{\theta}_d \) represent the median values of phases measured in the last 10 seconds at frequency \( f_j \) and common frequency \( f_d \) (default to 922.6525MHZ in this work). Fig.1(c) shows the results after phase unwrapping and phase smoothing.

2) Data Splitting and Resampling: We first split the collected long signal data into samples with width \( w \) (\( w = 5s \) in this work) and form the training set by them. Then, these samples are further divided into \( n \) (\( n = 10 \) in this work) non-overlapping timesteps \( X \) with time interval width \( \tau \) (\( \tau = 0.5s \) in this work). \( X = \{X_1, \ldots, X_{10}, \ldots, X_n\} \), \( X_t \) represents the data collected by \( k \) tags at timestep \( t \), which can be further divided into \( X_t = \{x_{t1}, x_{t2}, \ldots, x_{tk}\} \).

Due to the transmission characteristics of the RF signal, the RSSI and phase from each tag in the scenario cannot be read equally. We resample the input data to the same dimensions using linear interpolation.

3) Fast Fourier Transform: The raw RFID signals are a mixture of objects, too noisy to be directly understood and used. We use Fast Fourier Transform (FFT) to convert the time-domain data to the frequency domain for distinguishing the activity features from multiple objects.

4) Data Filtering: To select a more suitable filter in this scenario, our paper uses Gaussian Filter, Kalman Filter, Mean Filter, Median Filter, and Hampel Filter to process the raw
RFID signal. In the experimental section, we further compare the performance of the model with different filters.

IV. MODEL

In this section, we give a detailed description of the Deep-Multiple. The framework of DeepMultiple is shown in Fig. 2.

A. Individual ConvLayer

DeepMultiple is arranged according to $n$ timesteps as input. Since the structures of Individual ConvLayer at each timestep are the same, we focus on a single timestep with input $X_t = \{x_{t1}, x_{t2}, \ldots, x_{tk}\}$. Recall that $x_{ti}$ represents the signals from the $i$-th tag at timestep $t$ and the shape of $x_{ti}$ is $d \times 2f_j$, where $d$ present the tag measurement dimension, i.e. RSSI and phase, $f_j$ is the dimension of frequency domain. For each timestep, $x_{ti}$ is first fed into the dilated CNN with shape $(1, \text{conv}1)$ to extract the multi-scale activity features from different objects in the frequency domain. Then it uses a 2d conv with shape $(d, \text{conv}2)$ to merge features from different tags and a 2d conv with shape $(1, \text{conv}3)$ to further learn the high-level relationship, with the output $v_{ti}^{(1)}$. After each convolution operation, DeepMultiple uses ReLu as the activation function and applies Batch Normalization to reduce internal covariate shift and vanishing gradients.

B. Flatten and Merge Layer

In Flatten and Merge Layer, we flatten $v_{ti}^{(1)}$ in different channels into $v_{ti}^{(2)}$, and concat $k$ tags vector $\{v_{t1}^{(2)}, v_{t2}^{(2)}, \ldots, v_{tk}^{(2)}\}$ into a $k$-row matrix $V_t$, then we use conv2d with shape $(k, \text{conv}4)$ to learn the intrinsic interactions among all $k$ tags to generate the matrix $V_t'$, furthermore 2d filters with $(1, \text{conv}5)$ and $(1, \text{conv}6)$ are applied to learn the high-level relationships, last we flatten the result into $v_t'$ as the input to Transformer-based Encoder Variant Layer. Again, after each convolution layer, Batch Normalization and a ReLu activation are performed, and a MaxPool2d with stride $=2$ is applied to compress the dimensions in the last convolution layer.

C. Transformer-based Encoder Variant Layer

A large amount of training data should be used in transformer due to its complex structure, which is difficult to achieve in RFID filed. Previous research like [14] used simple RNN or GRU as the backbone network, but experiments have verified that these methods are not suitable for multi-object recognition without tags attachment. To this end, we optimize the vanilla Transformer-based Encoder structure and use sparse $\textit{ProbSparse}$ attention to effectively reduce the computational complexity and improve the recognition accuracy.

Since the vanilla self-attention mechanism uses atom operation, i.e. scaled dot product, causes time complexity and memory usage per layer to be $O(L^2)$. The use of the vanilla self-attention mechanism entails the computation of the scaled dot product of the input without discrimination, which will amplify the influence of noise in the data, hinder the performance of the model, and increase computational complexity. The $\textit{ProbSparse}$ self-attention mechanism proposed in [9] can be a good solution to this problem. It has shown that the distribution of self-attention probability is potentially sparse and achieves the $O(L \log L)$ time complexity and $O(L \log L)$ memory usage on dependency alignments. The $\textit{ProbSparse}$ self-attention mechanism is shown below.

$$A(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (4)$$

where $Q$ is a sparse matrix of the same size as $q$ and it only contains the Top-$\textit{u}$ queries under the sparsity measurement $\bar{M}(q, K)$ controlled by a constant sampling factor $c$, we set $u = c \cdot \ln L_Q$, and $A$ represents kernel smoother based on probability distribution. $\bar{M}(q, K)$ and $A$ can be described as:

$$\bar{M}(q_i, K) = \max_j \left\{ \frac{q_i k_j^T}{\sqrt{d_k}} \right\} - \frac{1}{L_k} \sum_{j=1}^{L_k} \frac{q_i k_j^T}{\sqrt{d_k}} \quad (5)$$

Under the long tail distribution, we randomly draw sample $U = L_K \ln L_Q$ dot-product pairs to get $\bar{M}(q_i, K)$. Then, we select Top-$\textit{u}$ from them as $Q$, so the $\textit{ProbSparse}$ self-attention time complexity and space complexity are $O(L \log L)$.

$$A(q_i, K, V) = \sum_j k(q_i, k_j) v_j = E_{p(k_j|q_i)} [v_j] \quad (6)$$

where $q_i$ stands for the $i$-th row in $Q$. $p(k_j|q_i) = k(q_i, k_i)/\sum_k k(q_i, k_i)$ and $k(q_i, k_i)$ selects the asymmetric exponential kernel $\exp \left(q_i k_j^T / \sqrt{d} \right)$. The self-attention and values are combined on the basis of the calculation of probabilities $p(k_j|q_i)$ to obtain the output.

We explain the Transformer-based Encoder Variant Layer with input $v_t'$ at timestep $t$ and start by applying the positional embedding [18] to introduce a concept of relative
order between the features extracted at different timesteps. Then, we multiple $v'_t$ with three different learnable matrices $W^Q, W^K, W^V$ to get the query, key, value matrices $Q, K, V$. The Top-10 important queries are selected according to (5) and the attention score is obtained by (6). Then we integrate attention score and input by skip-connection. Last, Layer Normalization is applied and then the normalized data are passed into the feedforward output layer to obtain the output $O = \{o_1, o_2, ..., o_t\}$.

D. Output Layer

The output after Transformer-based Encoder variant Layer is $O = \{o_1, o_2, ..., o_t\}$, we need a prediction activity $\hat{y}$, so a Linear layer, $\hat{y}_c = O \cdot A^T + b$, is used to map $O$ to $\hat{y}_c$ (where $c \in C$, $C$ is the set of all classes) and then normalize $\hat{y}_c$ by softmax, we select the max probability as follows.

$$P = \underset{c \in C}{\text{argmax}} \ (\text{softmax} (\hat{y}_c)) \tag{7}$$

We opt to use cross-entropy loss. It can be formulated as:

$$\mathcal{L} = - \sum_{c=1}^{N} y_c \log (P_c) \tag{8}$$

where $y_c$ is the indicator variable, $P_c$ is the probability that the predicted result belongs to class $c$, and $N$ is the number of class categories.

V. EXPERIMENT

A. DataSet Description

We conduct extensive experiments based on the dataset collected by five volunteers (three males and two females) in two typical indoor environments, a laboratory and an empty room to complex and simple multipath environments, respectively. In each environment, we deploy a single reader and antenna and then we attach a 3x3 tag array to a wall 1.5-2m above the ground. In our experiments, volunteers perform activities as shown in Fig.3. We tested ten activity scenarios of two people, as shown in Fig.4.

The hardware components of our experiment include an Impinj R700 reader, equipped with an ultrahigh-frequency UHF2599 antenna and SMARTAR揖 DogBone RFID tags.

We collected 3250 signal samples and name them RFAC DataSet, of which 80% were used as training sets, and 20% were used as test sets. To compare the impact of different filtering algorithms on model performance, we processed the raw data by different filters and constructed RFAC-Hampel, RFAC-Gaussian, RFAC-Kalman, RFAC-Median, and RFAC-Mean. In the subsequent experiments, the pre-processed dataset is applied to the baseline for performance comparison.

B. Experimental Details

In this section, we introduce the baseline, hyper-parameter tuning, and evaluation metrics used in this experiment.

Baseline: We chose five deep learning models used in the HAR field as a comparison, including the state-of-the-art models. For all the compared models, we only made minor adjustments to the shape of the input, and the model structures were implemented as provided by the authors. The CNN-stacked model, DeepConv [16], CNN-GRU, TagFree [8] and AtttnSense [17] are used to compare the performance with DeepMultiple.

Hyper-parameter tuning: We conducted a grid search for learning rate and weight decay, the learning search range was $\{1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}\}$, and the learning rate is finally set
to $1e^{-3}$, the weight decay search range was \{0.01, 0.1, 1\}, and the weight decay was finally selected to be 0.1. During the training, we used the early stopping to train and set the batch size to 64 then trained a total of 500 epochs. We use Adam optimizer with a default value and normalize the input by zero-mean.

**Metrics**: In this work, we use Weighted-F1 as the metric in the evaluation and the results are averaged over the 5 runs. The formula of Weighted-F1 is described as follows.

$$\text{Weighted - F1} = \sum_{i=1}^{C} w^{(i)} \times \frac{\text{Precision}^{(i)} \times \text{Recall}^{(i)}}{\text{Precision}^{(i)} + \text{Recall}^{(i)}}$$

where for a given class $i$, $w^{(i)}$ represents the proportion of this class in the total sample, Precision$^{(i)}$, Recall$^{(i)}$ are the precision and recall of the $i$-th class, respectively.

C. Numerical Analysis

1) **Performance of our model**: As shown in Fig.5, the Confusion Matrix shows the overall performance of DeepMultiple with the kalman filter at an interval width of 0.5s. The overall accuracy rate is 98.12% and the accuracy rate of all ten activities is above 96%. It can be found that our model can extract important features well in single reader-antenna tag non-attachment scenarios.

2) **Impact of time interval width and filter preprocessing**: In order to compare the impact of the time interval and filter preprocessing, we evaluated the performance of the model with different filters at different time interval widths. From Fig. 6, we can find that the model performs best when the time interval width is set to 0.5s. The highest Weighted-F1 score is 0.981 with the Kalman filter and the lowest Weighted-F1 score is 0.967 with the mean filter under this setting. In the subsequent experiments, the default is to compare performance with the Kalman filter at an interval width of 0.5s.

3) **Impact of phase calibration**: In order to verify that phase unwrapping and phase smoothing can also improve the accuracy of activity recognition, we further evaluate the performance of the model with calibration and no-calibration. As shown in Fig.7, we can find that our model has achieved higher recognition accuracy with calibration, which directly demonstrates the effectiveness of our phase calibration.

4) **Compared Algorithms**: In this section, DeepMultiple is compared with the baseline algorithm under the RFAC (D1), RFAC-Hampl (D2), RFAC-Gauss (D3), RFAC-Kalman (D4), RFAC-Median (D5) and RFAC-Mean (D6).

Through Table I, we find that the recognition accuracy of all models was improved after the filtering process, which proves the effectiveness of the filtering algorithm. In addition, we find that DeepMultiple always achieves the highest Weighted-F1 score regardless of filtering, which proves its strong robustness and generalization ability. In contrast, baselines without filtering have poor performance and cannot perform multi-object activity recognition in a complex multipath environment. There is no doubt that the performance of DeepMultiple is significantly better than the latest method in this field.

5) **Ablation Study**: To verify that the dilated convolution module and ProbSparse self-attention mechanism proposed by our model contribute to the effectiveness of activity recognition, we have performed two variants of our model, DeepMultiple_CConv using vanilla convolutions instead of dilated convolutions and DeepMultiple_CAtt using vanilla self-attention mechanism instead of ProbSparse self-attention. The result is shown in Fig.8.

By comparing the experimental performance of DeepMultiple, DeepMultiple_CConv and DeepMultiple_CAtt, it can be seen that the accuracy of the model using the dilated convolution has been improved in all ten activities. The overall recognition accuracy increased by 4.7%. The reason for this phenomenon is that vanilla convolution causes feature frag-
mentation when extracting features in the frequency domain, which can be solved by dilated convolution. Compared to the vanilla self-attention, using ProbSparse self-attention brings a 27.5% increase in global recognition accuracy. It can be concluded that the query sparsity of ProbSparse self-attention can effectively improve the performance of model recognition and prevent overfitting.

VI. CONCLUSION

In this paper, we propose DeepMultiple for multi-object activity recognition in complex multi-path environments. DeepMultiple not only applies the dilated convolution to feature extraction in the frequency domain, but it also reduces the calculation of the parameters with the selective ProbSparse attention and improves the recognition accuracy. Experimental results show that the accuracy of this model can reach 98.12%, which is superior to the state-of-the-art model in this field.

ACKNOWLEDGMENT

This paper is supported by the 2021 Fujian Foreign Cooperation Project(No. 2021I0001): Research on Human Behavior Recognition Based on RFID and Deep Learning; Horizontal project (Co-construction platform), 2023 Project of Xiamen University: Joint Laboratory of Public Safety and Artificial Intelligence(20233160C003); State Key Laboratory of Process Automation in Mining & Metallurgy, Beijing Key Laboratory of Process Automation in Mining & Metallurgy(No. BGRIMM-KZSKL-2022-14): Research and application of mine operator positioning based on RFID and deep learning; National Key R&D Program of China-Sub-project of Major Natural Disaster Monitoring, Early Warning and Prevention (No. 2020YFC1522604): Research on key technologies of comprehensive information application platform for cultural relic safety based on big data technology.

REFERENCES


A Dynamic Matching Time Strategy Based on Multi-Agent Reinforcement Learning in Ride-Hailing

Shuai Li\textsuperscript{1}, Bing Shi\textsuperscript{1,2}, Yaping Deng\textsuperscript{1}
\textsuperscript{1} School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China
\textsuperscript{2} Shenzhen Research Institute of Wuhan University of Technology, Shenzhen 518000, China
bingshi@whut.edu.cn

Abstract—For online ride-hailing platforms, choosing the right time to match idle vehicles with passengers is one of the most important factors affecting the platform’s profit. On one hand, vehicles and passengers arrive dynamically, and an appropriate delayed matching may generate a highly efficient matching result with more values. On the other hand, different regions may have different states of supply (vehicles) and demand (passengers), and the matching time should be different. At this moment, we need an efficient matching time strategy that takes into account matching time and regional differences to maximize the platform’s long-term profit. In this paper, we propose a dynamic matching time algorithm based on multi-agent reinforcement learning, which is called Multi-Region Differentiated Matching Decision. Firstly, we describe the order matching process and then model it as a decentralized partially observable Markov decision process (Dec-POMDP). Secondly, considering that there are regional differences in supply and demand, we divide the overall area based on historical data and propose an algorithm based on multi-agent reinforcement learning to realize multi-region differentiated dynamic matching. Finally, we conduct extensive experiments to evaluate our matching algorithm against benchmark algorithms in a real-world dataset. The experimental results show that our algorithm can outperform benchmark algorithms.

Index Terms—Ride-hailing, Delayed matching, Long-term profit, Multi-Agent Reinforcement learning

I. INTRODUCTION

Online ride-hailing has become one of the most important transportation ways in the modern cities. In the ride-hailing system, the platform needs to match idle vehicles with passengers efficiently, since this will significantly affect the platform’s profit and passengers’ riding experience. Currently, the ride-hailing platform usually matches the passengers with vehicles immediately when the riding orders are raised, such as [1], [2]. In fact, the immediate matching may cause inefficient matching results. For example, as shown in Figure 1(a), if the platform performs matching at time $t_1$, the cost of the platform and the waiting time of the passengers will increase due to the insufficient number of vehicles and passengers. In contrast, instead of immediate matching, the platform can collect more information about orders and idle vehicles to make more efficient matching decisions in the delayed matching, which is shown in Figure 1(b). However, the passengers may have a tolerance time for waiting for the service. Therefore, the platform needs to find an effective matching time. Furthermore, the riding requests and idle vehicles may change dynamically over the time, thus the platform needs to dynamically determine the matching time based on the arriving orders and idle vehicles.

Furthermore, in a city-wide area, different small subareas (called regions) may have different supplies (idle vehicles) and riding demands. In this case, it is inappropriate for the platform to set a uniform matching time for all regions. The platform needs to determine the matching time for each region respectively according to the supply and demand in that region. Moreover, the platform needs to maximize the long-
term profit, and thus needs to consider the impact of current decisions on the future matching.

Specifically, with the dynamic arrival of passengers and vehicles, the platform needs to choose the right matching time for regions with different supply and demand to maximize long-term profits. Note that it is difficult for the platform to set the matching time at any time since this will result in heavy computing load of keeping monitoring whether the matching information is enough. In this paper, we divide the whole time into several time slots. Therefore, instead of determining the matching at any time, in this paper we determine at what time slot to do the matching, i.e., to determine whether matching at the current time slot or not, to achieve dynamic matching. In summary, the main contributions of this work are as follows. Firstly, the current matching time decision affects future order matching, and each region can only make decisions based on its observed supply and demand information, so this is a sequential decision problem, which we model as a decentralized partially observable Markov decision process (Dec-POMDP). Secondly, we divide the whole area into several non-overlapping regions according to the historical order data to realize multi-region differentiated dynamic matching. Thirdly, we design a multi-agent reinforcement learning based algorithm (MRDMD) to determine whether matching at the current time slot or not in order to maximize the long-term profit of the platform. Finally, we conduct extensive experiments to evaluate the proposed algorithm based on a real-world dataset. We demonstrate that using differentiated matching time in different regions can significantly increase the platform’s long-term profit.

The rest of this paper is organized as follows. In Section II we introduce the related work. In Section III we describe the basic settings and define the problem. We introduce the multi-region differentiated matching decision algorithm in Section IV. Finally, we give experimental analysis in Section V and conclude the paper in Section VI.

II. RELATED WORK

The order matching problem in ride-hailing is widely studied. There exist a lot of works that consider how to maximize the platform’s profit. Li et al. \cite{3} propose a non-centralized order matching approach, where vehicles are viewed as agents, and multiple agents work together to maximize the overall profit. Shi et al. \cite{4} propose two order matching mechanisms based on truthful auction, where drivers bid for orders published by the platform and make profits in order to maximize the social welfare of drivers and platform.

There also exist a number of works about maximizing the number of completed orders in order matching problem. Garaix et al. \cite{5} propose direct and iterative algorithms to solve the order matching problem and maximize the number of completed orders. Furthermore, Holler et al. \cite{1} propose a deep reinforcement learning approach by combining deep learning. They treat vehicles as independent agents and perform order matching from the perspective of centralized platform dispatching, thus maximizing the number of completed orders.

Moreover, how to minimize vehicle travel distance is also investigated in the related works. Liao et al. \cite{6} propose a nearest matching algorithm to match the order with the vehicle closest to it. Duan et al. \cite{7} propose an algorithm that can gradually expand the visible range of orders, which can effectively reduce the allocation time of orders and maintain a low travel distance.

The above works usually consider immediate matching between vehicles with passengers. Some works, \cite{3, 6}, consider the delayed matching by using the cumulative information. Qin et al. \cite{9} find that dynamic matching can effectively reduce passenger’s waiting time compared to immediate matching. However, they do not take into account that in different regions, the platform has different supply and demand information, therefore the matching time can be set differently.

To the best of our knowledge, existing works did not consider that different regions have different supply and demand with dynamic changes over time, and thus they cannot dynamically determine the matching time for each region for each time to maximize the profit of the platform. In this paper, we consider the above factors and design a dynamic matching decision algorithm MRDMD for multiple regions to maximize the long-term profit of the platform.

III. BASIC SETTINGS

In this section, we first describe how the ride-hailing system works, and then we give the basic settings of the order, vehicle and platform’s profit. Furthermore, we describe the problem we intend to solve in this paper.

Figure 2 shows how the ride-hailing system works. When a passenger rises a trip order, first this order will be collected by the platform into the buffer pool. Second, when the size of the buffer pool is suitable, the platform will match those orders with idle vehicles. Then the vehicle that receives the allocated order will transport the passenger to the designated location and charge the appropriate fee.

We divide the whole time into several time slots $T = \{1, 2, ..., T\}$ and consider a 2-D area $L = \{1, 2, ..., L\}$. The relevant settings are given below.

**Definition 1 (Order):** An order $o_i \in O$ is a travel request made by a passenger through a smart device, which can appear at any time slot but can only be served by a maximum of one vehicle. An order $o_i$ is expressed as a tuple $o_i = (o_{i, r}, o_{i, s}, o_{i, d}, o_{i, c}, o_{i, w}, o_{i, s})$, the specific meaning of these elements is as follows:
• \( t_{o_l} \) is the time when the order \( o_l \) is submitted.
• \( o_{g_{o_l}} \) is the passenger’s pick-up location, consisting of latitude and longitude.
• \( des_{o_l} \) is the destination of the order, consisting of latitude and longitude.
• \( p_{o_l} \) is the price of the order.
• \( c_{o_l} \) is the cost of the order, which is determined by the distance traveled and the unit cost of the vehicle.
• \( w_{o_l} \) is the maximum tolerance time for the passenger.
• \( s_{o_l} \) is a status identifier to indicate the status of the order, including pending, completed, and invalid.

**Definition 2 (Vehicle):** The vehicle \( v_j \in V \) is represented as a tuple \( v_j = (loc_{v_j}, s_{v_j}) \), the specific meaning of these two elements is as follows:
- \( loc_{v_j} \) represents the current location of the vehicle, consisting of latitude and longitude.
- \( s_{v_j} \) indicates the current state of the vehicle, to show whether the vehicle is idle or busy.

We assume that the vehicles are owned by the platform and the vehicles will follow the platform’s dispatch to serve passengers, which simplifies the management of the platform and also improves efficiency [8].

**Definition 3 (Platform’s Profit):** The platform’s profit is equal to the value of all completed orders minus the cost of vehicles.

\[
EP = \sum_{l=1}^{L} \sum_{t=1}^{T} \sum_{i=1}^{O_l^t} (p_{o_i} - c_{o_i})
\]  
(1)

\( O_l^t \) is the completed order set in the region \( l \) at time slot \( t \), \( p_{o_i} \) and \( c_{o_i} \) are the value and cost, respectively, of the \( i \)th completed order in region \( l \) at time slot \( t \).

Based on the above settings, in this paper, we consider a ride-hailing system where vehicles and orders arrive dynamically and passengers have a maximum tolerance time. We want to dynamically determine the matching time in multiple regions to maximize the long-term profit \( EP \). To achieve this goal, we design a Multi-Region Differentiated Matching Decision (MRDMD) algorithm, which dynamically determines the matching time for each region. Note that as we discussed in Section [1] it is difficult for the platform to set the matching time at any time. Therefore, in this paper we determine at what time slot the platform will do the matching, i.e., to determine whether matching at the current time slot or not, to achieve dynamic matching.

**IV. The Algorithm**

In this section, we consider the decentralized partially observable Markov decision process (Dec-POMDP), which describes the multi-region differentiated order matching process, and then propose a multi-region differentiated matching decision (MRDMD) algorithm to maximize the profit of the platform.

The Dec-POMDP in our multi-regional cooperation scenario can be represented as a tuple: \( G = < S, A, P, r, Z, \emptyset, n, \gamma > \). Where \( s \in S \) represents the area’s supply and demand information. At each time slot, each region \( l \) obtains its observation \( z_l^t \in Z \) from the environment using the observation function \( O \). Based on the observation, each region chooses an action \( a_l^t \in A \) and all the actions are combined to form a joint action \( a \in A^L \). After this joint action is performed, the environment state is transferred to the new one according to the state transition function \( P(s_{t+1}|s_t, a_t) : S \times A^L \times S \rightarrow [0, 1] \). \( r \) represents the reward function, which is shared by all the regions, \( n \) represents the number of regions, and \( \gamma \in [0, 1) \) is a discount factor that decreases the impact of the past reward. We describe the details of the observation space, action space, and reward function below.

**Observation Space:** \( z_l^t = ((v_1^t, o_{1}^t),(v_2^t, o_{2}^t),\ldots,(v_l^t, o_{l}^t)) \in Z^L \), where \( o_{1}^t, o_{l}^t \) represent the number of vehicles and orders respectively in region \( l \) at time slot \( t \).

**Action Space:** \( a_l^t = (a_1^l, a_2^l, \ldots, a_l^l) \in A^L \), where \( a_1^l = 0 \) and \( a_l^l = 1 \) represent delayed matching and matching in region \( l \) at time slot \( t \), respectively. Note that the unmatched orders and vehicles will transfer to next time slot.

**Reward Function:** At each time slot \( t \), each region \( l \) takes the action \( a_l^t \in \{0, 1\} \), and then get a reward \( r_l^t = \sum_{i=1}^{O_l^t} (p_{o_i} - c_{o_i}) \) from environment. The global reward \( r_t = \sum_{l=1}^{L} r_l^t \) is accounted for all regions in order to obtain the maximum overall profit.

There are multiple regions in our scenario, each requiring individual decisions and ensuring that the long-term profit of the platform is maximized. Therefore, we use QMIX [10] to design MRDMD, which is based on decentralized training and distributed execution and is widely used in various multi-agent environments. In our algorithm, there exist three types of networks, the regional network \( Q_l \), the mixed network \( Q_{tot} \), and the hypernetwork. The regional network performs an action based on individual observation \( z_l^t \) and previous action \( a_{l-1}^t \) and outputs regional action values \( Q_l(\tau^t, a_l^t) \), where \( \tau \) is the observation-action history, maintained by each region to perceive the dynamic changes of region information. The mixed network has weight parameters produced by the hypernetwork consisting of multiple linear layers and receives as input the state \( s \in S \). This network receives the action values \( Q_l(\tau^l, a_l^l) \) of all regions and outputs the overall action values \( Q_{tot}(\tau, a) \). In summary, MRDMD needs to ensure that equation [2] is satisfied, which means that each region can greedily choose matching action based on the regional network solely. The MRDMD algorithm is shown in Algorithm [1].

\[
\text{argmax}_a Q_{tot}(\tau, a) = \begin{cases} 
\text{argmax}_{a_1} Q_1(\tau_1, a_1) \\
\vdots \\
\text{argmax}_{a_L} Q_L(\tau_L, a_L)
\end{cases}
\]  
(2)

Algorithm [1] takes the number of vehicles and regions as input. The training will take \( K \) rounds (line 4), each lasting for \( T \) time slots (line 7). At the beginning of the training, each agent acquires an initial observation state (line 6). During training, each agent first gets the action value by using the evaluation network \( Q(\theta) \) (line 8). After that, these actions will be collected to get the joint action \( a \) (line 9). For each region, if
When the memory size is greater than a threshold, we will store this information in the memory pool (line 19). When a matching action is selected, it will use Kuhn-Munkres [11] algorithm to find the most profitable matching combination of the current vehicles and passengers (line 12). When no matching is made, the current vehicle and order information is transferred into the next round (line 15). After the matching is completed, we can calculate the real-time reward \( r \) of the matching action and get the next state \( s' \) (line 18), and then this information will be stored in the memory pool (line 19). When the memory size is greater than a threshold, we will randomly sample some records from the memory pool for learning (lines 20-23). After continuous repeated learning and training, a convergent matching strategy that can maximize the platform’s long-term profit is obtained.

V. EXPERIMENTAL ANALYSIS

In this section, we run extensive experiments to evaluate our algorithm. The dataset used in this paper is provided by DiDi, from which we select the data from 13:00 to 15:00 on weekends, which contains the initiation time, the end time, the start location and destination of the order, and also the price obtained by completing the order. Similar to [12]–[14], we divide the area into four regions with different supply and demand based on the number of orders, which is shown in Figure 3.

Vehicles and passengers arrive dynamically, and their activity range is bounded according to the maximum range of all orders in the dataset. The initial state of each vehicle is idle. The number of vehicles gradually increases from 500 to 2000, with each increase of 500 vehicles. The period is 13:00-15:00, and we use \( \Delta t = 10s \) as the length of time slot. There will be 720 time slots during the entire period. We set the vehicle's unit driving cost to 1 CNY/km. Each order has a maximum tolerance time \( w \), and we assume it is independently and identically drawn from a uniform distribution within \([1, 30]\). When a passenger waits longer than the tolerance time, the order will be cancelled. In this experiment, we assume that the vehicle only serves orders within the same region at the current time slot, but after the service is completed, the vehicle can move to other regions. The specific experimental parameters are shown in Table I.

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time period ( T )</td>
<td>13:00-15:00</td>
</tr>
<tr>
<td>Length of time slot ( \Delta t )</td>
<td>10</td>
</tr>
<tr>
<td>Number of vehicles</td>
<td>500, 1000, 1500, 2000</td>
</tr>
<tr>
<td>Maximum tolerance time ( w )</td>
<td>( U(1, 30) )</td>
</tr>
<tr>
<td>Number of regions ( L )</td>
<td>4</td>
</tr>
<tr>
<td>Unit driving cost</td>
<td>1</td>
</tr>
</tbody>
</table>

TABLE I

**EXPERIMENTAL PARAMETERS**

- Time period: 13:00-15:00
- Length of time slot: 10
- Number of vehicles: [500, 1000, 1500, 2000]
- Maximum tolerance time \( w \): \( U(1, 30) \)
- Number of regions \( L \): 4
- Unit driving cost: 1
A. Benchmark Approaches and Metrics

We will use the following benchmark approaches and metrics to evaluate our proposed algorithm.

Multi-Region Restricted Q-Learning (MRRQL). The RQL algorithm [15] sets a maximum matching time interval $[a, b]$, and then uses reinforcement learning to continuously adjust the size of this interval to control the matching time. We modify the RQL algorithm to fit our problem, where each region performs the RQL algorithm independently, which is called the MRRQL.

Multi-Region GREEDY (MRGREEDY). Tong et al. [2] find that GREEDY algorithm can still achieve very competitive results in most cases. We modify the GREEDY algorithm to fit our problem, where each region performs matching independently at each time slot using the GREEDY algorithm, and priority is given to match the highest value orders with idle vehicles, which is called the MRRGreedy.

Multi-Region UNIFORM (MRUNIFORM). The UNIFORM algorithm [16] is a commonly used comparison algorithm, which will do the matching for every $n$ time slots. We modify the UNIFORM algorithm to increase its dynamic matching property, with a half probability of matching at the current time slot and a half probability of not making a match, which is called the MRUNIFORM.

In order to evaluate the performance of MRDMD, we consider the following metrics.

- **Total platform’s profit.** The total platform’s profit refers to the sum of the profit of all vehicles.
- **Order response rate.** Order response rate is the ratio of completed orders to total orders.
- **Pick-up distance.** Pick-up distance is the distance from the vehicle’s current location to the order initiation location after the vehicle matches the order.
- **Average extra distance per order.** The average extra distance per order is the average distance an empty car needs to travel to complete an order.

B. Experimental Results

In the experiment, we increase the number of vehicles from 500 to 2000 with step size 500, and the experimental results are shown below.

To prove that our algorithm is effective when combined with the region division, we conduct experiments on the whole area and multiple regions respectively to compare the profit of all algorithms. We use MR to represent an algorithm that combined with region division, e.g., MRDMD and DMD are the same algorithm running on multiple regions and the whole area, respectively. From Figure 4(a) and Figure 4(b), we can see that MRDMD can make higher profits than DMD, and similarly MRRQL can make higher profits than RQL, which means that after the area is divided, the reinforcement learning based algorithm can sense the differentiated information of each region and make independent and effective matching decisions. In contrast, from Figure 4(c) and Figure 4(d), we can find that MRRGreedy and MRUNIFORM do not perform well as that GREEDY and UNIFORM do.

Next, we will evaluate the proposed algorithm with multiple regions against the benchmark approaches.

**Total Platform’s Profit:** The total platform’s profit of the four algorithms is shown in Figure 5. As the number of vehicles increases, we find that the profit of all four algorithms increase. We also find that MRDMD performs better than the other three algorithms. MRRQL performs better than the other two algorithms. In more detail, we find that the algorithm with dynamic delayed matching time (MRDMD, MRRQL) brings more profit to the platform compared to the algorithms with immediate matching (MRGREEDY) and dynamic random matching (MRUNIFORM).

![Platform’s profit between multiple regions and overall area.](image)

**Order Response Rate:** As we can see in Figure 5 when the idle vehicles are insufficient, the order response rate of the four algorithms are very similar, where MRGREEDY has a slight advantage. This is because the MRGREEDY matches orders with vehicles at each time slot, and therefore has a better order response rate. When the number of vehicles is greater than 1500, MRDMD still performs the best, followed by MRRQL, MRGREEDY and MRUNIFORM.

**Pick-up Distance:** From Figure 6 we can see that the pick-up distance of the four algorithms increases when the number of vehicles is increased to 1500. When the number of vehicles is greater than 1500, the pick-up distance decreases for MRDMD and MRRQL, while the distance still increases for MRGREEDY and MRUNIFORM. This may be because MRDMD and MRRQL can sense the dynamic changes of the regional state about vehicles and orders, and when there

![Platform’s profit of each vehicle.](image)
exist a large number of vehicles, these two algorithms can accumulate enough information about orders and vehicles to make efficient matching decisions, which may decrease pick-up distance. Finally, we still find that MRDMD performs the best.

**Average Extra Distance Per Order:** From Figure 8, we can see that the average extra distance per order for the four algorithms decreases as the number of vehicles increases and MRDMD has better performance. This is because more vehicles bring more matching information, and for an order, a closer vehicle can be selected for matching. Therefore the average distance to complete an order decreases.

![Fig. 6. Order response rate.](image)

![Fig. 7. Pick-up distance.](image)

![Fig. 8. Average extra distance per order.](image)

VI. CONCLUSION

In this paper, we propose a multi-region differentiated matching decision algorithm (MRDMD) based on multi-agent reinforcement learning by considering the real-time supply and demand status of different regions in order to maximize the ride-hailing platform’s profit. In order to evaluate the effectiveness of MRDMD, we run experimental analysis based on the real DiDi dataset against three typical benchmark algorithms. The experimental results show that the proposed algorithm can outperform other algorithms. We find that our algorithm with dynamic matching time according to the supply and demand status of each region can bring higher long-term profit and serve more orders. Our analysis can also provide useful insights for designing the realistic matching time strategy for ride-hailing platforms.

**ACKNOWLEDGEMENT**

This paper was funded by the Shenzhen Fundamental Research Program (Grant No.JCYJ20190809175613332), the Humanity and Social Science Youth Research Foundation of Ministry of Education (Grant No.19YJC790111) and the Philosophy and Social Science Post-Foundation of Ministry of Education (Grant No.18JHQ060).

**REFERENCES**


mTrader: A Multi-Scale Signal Optimization Deep Reinforcement Learning Framework for Financial Trading

Zhennan Chen, Zhicheng Zhang, Pengfei Li, Lingyue Wei, Shibo Feng, Fan Lin*
School of Informatics, Xiamen University, Xiamen, China
{znchen, zczhang, lpfei, 31520211154091, fengshibo1024}@stu.xmu.edu.cn, *iamafan@xmu.edu.cn

Abstract—It is universally acknowledged that the financial trading is a thorny issue in time-series scenarios. On the one hand, due to the great randomness and instability in financial markets, the existing machine learning methods are inadequate for modeling high-frequency financial data. On the other hand, it remains a challenge to identify the validity of transaction actions to avoid high fees. To address these issues, we propose a novel trading framework, namely mTrader, to offer suitable trading strategies automatically. We creatively design a multi-scale signal matrix to describe the temporal trends of markets. On this basis, Vector Quantized Variational AutoEncoder (VQ-VAE) was introduced to capture discrete latent variables. In addition, an offline Action Optimizer (AO) based on Proximal Policy Optimization (PPO) could help filter out sub-optimal trading action. Extensive experiments have shown that our model achieves state-of-the-art performance on many popular stock markets.

Keywords—Multi-Scale, VQ-VAE, Proximal Policy Optimization (PPO), Financial Trading

I. INTRODUCTION

Financial trading aims to obtain optimal returns while avoiding market fluctuations. During the past few decades, traders have proposed many trading strategies [4], [14], but poor usability resulted in these traditional strategies only helping traders to trade at a fixed time or in a fixed market. Due to the huge demand for funds, new intelligent trading algorithms have been proposed to help traders adapt to different market changes.

Processing financial transaction data is considered to be one of the most challenging tasks due to the evolutionary and nonlinear nature of financial markets. As the global financial market gradually becomes large and complex, the number of statistical models [2] and sophisticated models [21] has risen during the recent years. However, traditional models show insufficient processing ability while facing with large input states. Thus methods based on Deep Reinforcement Learning (DRL) have become the preferred choice. For instance, deep Q-network (DQN) [12] is proposed to learn policies from high-dimensional inputs, in order to suit the situation approaching real-world complexity. Jiang et al. [5] use the model-free Deep Deterministic Policy Gradient (DDPG) to deal the portfolio management problem with a deep learning solution. Schulman et al. [16] present Proximal Policy Optimization (PPO) to hedge a portfolio of derivatives against market frictions. Despite numerous attempts to apply DRL in financial markets, unpredictable factors can still affect prediction results, leading to highly non-stationary time series that make it challenging to observe real markets with extreme volatility. Additionally, the input states often contain a large amount of noise that interferes with the selection of subsequent actions.

To address these challenges, we propose a novel deep reinforcement learning framework, mTrader, which models the financial trading process as a Partially Observable Markov Decision Process (POMDP) [6]. In our framework, we design a multi-scale signal optimization approach to describe features of different markets using a sparse signal matrix, which helps to capture the temporal trends of markets. We then employ a Vector Quantized Variational AutoEncoder (VQ-VAE) [19] to observe current market states precisely. Moreover, we introduce an offline Action Optimizer (AO) based on Proximal Policy Optimization (PPO) to filter out sub-optimal trading actions, enabling the framework to maximize returns even in the presence of market fluctuations. Our proposed framework achieves state-of-the-art performance on various popular stock markets. Our main contributions are summarized as follows:

- We proposed mTrader, a novel multi-scale signal optimization framework, which can automatically make suitable decisions during financial transaction process.
- To extract meaningful features, we use a multi-scale sparse signal matrix for features expression and use VQ-VAE for temporal modeling of high frequency financial data. To our knowledge, this is the first effective application of VQ-VAE in this field. We introduce an offline Action Optimizer (AO) in order to identify the validity and rationality of certain transaction behaviors.
- We perform extensive experiments to demonstrate the advantages of our framework over other algorithms and its versatility in different markets.

II. RELATED WORK

1) Feature Extraction: It is crucial to extract high-level features from financial data for accurate prediction of stock prices. Tsantekidis et al. [18] described a deep learning method...
based on convolutional neural networks (CNN) to predict the price movement of stocks. Liu et al. [9] used LSTM recurrent neural network to extract feature values, established a corresponding stock trading prediction model by analyzing stock data. All in all, it turns out that CNN and LSTM are undoubtedly the most popular feature extraction methods. Since financial data is the sequence at regular intervals of time and LSTM has superior performance in processing series data, using LSTM for feature extraction is likely to be a more suitable choice for financial data analysis and stock price prediction.

2) Deep Reinforcement Learning: Owing to the outstanding ability of DRL to solve complex continuous decision-making problems, applications in financial market transactions have emerged one after another. Some researchers have used critical-only approaches, usually DQN [11] and its variants, to solve financial transaction tasks. Other researchers have focused on actor-only approaches, where Moody et al. [13] creatively proposed an adaptive algorithm using recurrent reinforcement learning (RRL) to optimize risk for better return on investment. On the other hand, some researchers have used actor-critic approach. For example, Yang et al. [20] utilized three algorithms to obtain an ensemble policy that can robustly adapt to complex environments better, and is more suitable for financial transaction tasks.

III. PROBLEM FORMULATION

In this section, we first clarify the full definition of POMDP then formally introduce the financial trading problem in DRL.

A. POMDP

The POMDP is a realistic generalization of a Markov Decision Process (MDP) for the financial trading problem. An MDP is a 5-tuple \( M = (S, A, P_T, R_T, \gamma) \). \( S \) is a finite set of states. \( A \) is a finite set of actions. \( P_T : S \times A \times S \rightarrow [0,1] \) denotes the state transition function. \( R_T : S \times A \rightarrow \mathbb{R} \) is the reward function, where \( \mathbb{R} \) is a continuous set of possible rewards. \( \gamma \in [0,1) \) is the discount factor. The goal of this process is to learn a policy \( \pi : S \rightarrow A \) that maximizes the expected discounted cumulative reward \( \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R_t \mid \pi \right] \). The action value function \( \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R_t \mid \pi \right] \) is used to evaluate policy \( \pi \).

In our financial trading framework, \( \mathcal{O} = \{Z, \mathcal{R}\} \) is a set of observations that contains our constructed feature encoding \( Z \) and reward \( \mathcal{R} \). \( B : S \times A \times O \rightarrow [0,1] \) is the observation transition function. \( \mathcal{O} \) and \( \mathcal{B} \) are the remaining components of POMDP.

At each timestep \( t \), the agent takes action \( a_t \in A \) according to the current environment \( s_t \in \mathcal{S} \). Then get \( s_{t+1} \) through the state transition function \( P_T(s_{t+1} \mid s_t, a_t) \). At the same time as the environment \( s_{t+1} \) is obtained, the agent will get \( o_{t+1} \in \mathcal{O} \) through the observation transition function \( B(a_{t+1} \mid s_{t+1}, a_t) \).

B. State

Previously, researchers mostly used the primitive properties of the raw data to represent financial market signals. In this paper, we perform state augmentation using technical indicators recognized by financial scientists as helpful in observing the market, and further process it into a multi-scale sparse signal matrix at time \( t \). After encoding, the state is got and input into the DRL model.

C. Action

Like real market transactions, our action space is discrete. The action of period \( t \) is \( a_t = \{a_b, a_s, a_h\} \), where \( a_b \) means to buy, \( a_s \) means to sell, and \( a_h \) means no action to buy or sell. The amount of each action is fixed.

D. Reward

The reward of period \( t \) is the Sharpe ratio. Sharpe ratio (Sr), a measure that considers both return and risk, is used to adjust past or expect future performance of a portfolio for excess risk taken by traders, defined as the average of the risk-free return by its deviation:

\[
Sr = \frac{\mathbb{E}(R_p - R_f)}{\sigma(r)},
\]

where \( R_p \) is the rate of return, \( R_f \) is the risk-free rate, \( \sigma(r) \) is the standard deviation of returns. To accurately measure trading action, Sr is calculated over the past \( n \) sliding windows.

E. Assumptions

In our model, there are two general assumptions: 1) Taking action will not have an impact on the market. Markets for financial transactions are resilient, and the impact of agency investment on the market will recover before the next step. 2) Immediate execution of the market. We assume that the agent’s behavior can be implemented directly, that is, there is no delay in each investment.

IV. PROPOSED METHOD

The overview architecture of mTrader is shown in Figure 1. In this section, we will introduce our model, mTrader, which is designed to solve decision-making problems in financial trading.

A. Multi-Scale Signal Modeling

1) Multi-Scale Signal Metrics: The special mode of the sparse signal matrix makes it have strong feature expression ability in financial data. According to the results of past research [1], [8], we have selected 32 indicators that can reflect market trends best, such as RSI, MACD, KDJ etc. In order to eliminate irrelevant or strongly correlated features among the 32 technical indicators while expanding the data expression ability to the greatest extent, we use Least Absolute Shrinkage and Selection Operator Algorithm (LASSO) [17] for further feature selection to ensure that each indicator can contribute to the final result. In this operator, features with negative or zero coefficient values can be removed from the feature subset, and
finally the most common feature is taken as the most important feature. Its estimated value is defined as follows:

\[
\hat{\beta}_\alpha = \arg \min_{\beta} \left( \frac{1}{n} \sum_{i=1}^{n} (y_i - \sum_{j} x_{ij} \beta_j)^2 + \alpha \sum_{j=1}^{p} |\beta_j| \right),
\]

(2)

where \( \beta \in \mathbb{R}^p \) and represents the coefficient of each feature, \( \alpha \) is an adjustable penalty parameter, and some inactive components of \( \beta_\alpha \) can be precisely set to 0.

The input raw data \( X_t = \{X_O, X_H, X_L, X_C\} = [x_{t1}, ..., x_{tk}] \) includes the opening price indicator \( X_O \), the highest price indicator \( X_H \), the lowest price indicator \( X_L \) and the closing price indicator \( X_C \). We judge each technical index after feature screening, and treat \( X_t \) as a sparse signal matrix \( G_t \) containing only -1, 0, 1. We introduce multi-scale information. For \( M_t \) of sliding window, \( M_t^s \) describes the strong correlation of transactions, \( M_t^m \) focuses on descriptiveness, and \( M_t^l \) shows trend. \( M_t \) is defined as follows:

\[
M_t = \begin{bmatrix}
\vdots & \vdots & \vdots \\
(M_t^s)_{j} & \cdots & (M_t^m)_{j} & \cdots & (M_t^l)_{j} \\
\vdots & \vdots & \vdots 
\end{bmatrix}.
\]

(3)

2) VQ-VAE Encoder: VQ-VAE [19] is a self-supervised model that includes an encoder that maps observations to a sequence of discrete latent variables, and a decoder that reconstructs observations from these discrete variables. VQ-VAE is more effective than traditional encoding methods for modeling financial discrete signals because it maintains the consistency between the input and output in a discrete format, allowing it to capture the sequential nature of financial data and model complex patterns more effectively.

VQ-VAE includes three parts: encoder, vector quantization codebooks, and decoder. We define \( E^V = [e_1, ..., e_K] \in \mathbb{R}^{K \times D} \) as a collection of codebook vectors. To transform the multi-scale sparse signal matrix \( M_t^s \) into a discrete variable, the encoder first produces intermediate continuous representation \( Z_e^V \in \mathbb{R}^D \). Later on, we scan through all codebooks to find which codebook has a minimum distance between \( Z_e^V \) and a vector in \( E^V \). After the closest codebook index \( k \in \{1, ..., K\} \) is found, we substitute latent variable \( Z_e^V \) with nearest codebook vector \( e_k \). Then, the decoder uses codebook vector \( e_k \) to reconstruct input feature \( M_t \). The definitions are as follows:

\[
Z_e^V = C(M_t),
\]

(4)

\[
Z_t^V = \text{VectorQuantization}(Z_e^V) = e_k,
\]

(5)

\[
M_t = D(Z_t^V).
\]

(6)

Given input reconstruction across all time step \( \forall t \in [1, T] \), the VQ-VAE is trained using the following objective:

\[
\mathcal{L} = ||M_t - \hat{M}_t||^2 + ||sg[Z_e^V] - Z_t^V||^2 + \varphi||Z_t^V - sg[Z_e^V]||^2,
\]

(7)

where \( sg \) refers to a stop-gradient. The hyper-parameter \( \varphi \) scales the commitment loss. The first term is used to measure the reconstruction loss between the original input \( M_t \) and the reconstructed \( \hat{M}_t \). The second and third term minimize the distance between intermediate representation \( Z_e^V \) and \( Z_t^V \).

B. Action Optimizer based PPO

The offline Action Optimizer (AO) will further analyze the actions obtained from PPO, and discard invalid actions, thereby avoiding losses or high fees.
1) Proximal Policy Optimization (PPO): In our trading framework, we use the the PPO algorithm [16], which is an actor-critic method with well-proven performance. To optimize the policy, PPO learns the policy through iteratively sampled data interacting with the environment and optimizes the surrogate objective function using stochastic gradient ascent. It alternates between the sampled data and optimizing the surrogate objective function defined as follows,

\[
L^C(\theta) = \mathbb{E}_t \left[ \min \left( r_t(\theta) \hat{A}_t, \text{clip} \left( r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right],
\]

\[
L^{KL}(\theta) = \mathbb{E}_t \left[ (r_t(\theta) \hat{A}_t) - \beta KL (\pi_\theta || \pi_\theta) \right],
\]

where \( \theta \) is the policy parameter. The updated \( \beta \) is used for the next policy update. \([1 - \epsilon, 1 + \epsilon]\) is the probability ratio of the clipping region.

\[
r_t(\theta) = \frac{\pi_\theta(a_t, s_t)}{\pi_\theta(a_t, s_t)},
\]

where \( \pi \) is the policy. PPO uses importance sampling to prevent the new policy going far away from the old policy. \( \hat{A}_t \) is the advantage estimator,

\[
\hat{A}_t = \delta_t + (\gamma \lambda) \delta_{t+1} + \cdots + (\gamma \lambda)^{T-t+1} \delta_{T-1},
\]

where \( T \) is the total timesteps in an episode, \( \gamma \) is the discount factor, and \( \lambda \) is a hyperparameter to control the bias-variance trade-off. \( \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \).

In addition, we introduce LSTM [3] into PPO, hoping that the reinforcement learning algorithm can clarify the relationship between high-frequency data. Compared with modeling in the original data, this method has stronger expressible ability and weaker noise features, which is particularly critical for the subsequent selection of actions in reinforcement learning.

2) Offline Action Optimizer (AO): In order to keep agents’ actions within bounds at all times and trade in the most efficient way, we propose an efficient offline Action Optimizer (AO) to optimize policies with boundaries. Inspired by methods for solving reinforcement learning via convex constraints [10], our AO allows for effective policy optimization using reinforcement learning by constraining the space of possible actions.

We define the AO as the boundaries, \( P_s, P_b \) are the upper and lower boundaries, which are related to buying and selling, respectively. \( P_{h1}, P_{h2} \) are the limits used to control the hold action,

\[
P_s(i + 1) = \sum_{i=1}^{n} \frac{p_c(i + 1) - p_c(i)}{p_c(i)} (1 + m_1),
\]

\[
P_b(i + 1) = \sum_{i=1}^{n} \frac{p_c(i + 1) - p_c(i)}{p_c(i)} (1 - m_1),
\]

\[
P_{h1}(i + 1) = \sum_{i=1}^{n} \frac{p_c(i + 1) - p_c(i)}{p_c(i)} (1 + m_2),
\]

\[
P_{h2}(i + 1) = \sum_{i=1}^{n} \frac{p_c(i + 1) - p_c(i)}{p_c(i)} (1 - m_2),
\]

where \( m_1 \) and \( m_2 \) are margin parameters within window size \( n \). \( p_c(i) \) is the closing price on day \( i \). Once the action given by the model exceeds the bounds set by AO, it means that action needs to be optimized. If the \( P_s \) boundary is crossed upwards, the trading action needs to be optimized to sell. In the same way, if the \( P_b \) boundary is crossed down, the trade action needs to be optimized to buy. The middle boundary is set to prevent frequent invalid transactions. If the action given by the model cannot break through the \( P_{h1}, P_{h2} \) boundary, we hope to continue to hold it, otherwise the cost of handling fees will be huge.

V. Experiments

In this section, we back-test mTrader in three completely different markets, define evaluation metrics, compare with other baselines, and evaluate its performance.

A. Experimental Setup

1) Datasets and Baselines: We selected the most representative five-minute datasets from three different markets: Bitcoin, Tesla, and Contemporary Amperex Technology Co. Limited (CATL) to verify the generalization and robustness of our model, as shown in Figure 2. The training and test datasets’ period division is illustrated in Table I. The three datasets represent the cryptocurrency market, and it should be noted that their overall trends are completely different, which can better test the model’s capability. We compared our proposed mTrader with three baselines, DQN [12], DDGP [5], and PPO [16].

![Fig. 2: Closing price variation of Bitcoin, Tesla and CATL.](image)

2) Metrics: Four commonly used metrics are used in our experiments, including Annualized Sharpe Ratio (ASr), Annualized Volatility (AVol), Maximum Drawdown (MDD), Total Return Rate (Tr). The ASr is the average return earned in excess of the risk-free rate per unit of volatility or total risk. The AVol is a statistical measure of returns. In general, greater volatility means greater uncertainty. MDD denotes the biggest loss from a peak to a trough. Tr is the most important objective of financial trading to make profits. Therefore, the final accumulative value reflects the performance of models.
B. Experiment Results

The performances of all the evaluated methods in the three testing datasets are summarized in Table II. Figure 3 shows the cumulative returns across three datasets.

![Figure 3: Cumulative return rates on Bitcoin, Tesla and CATL.](image)

From Table II, our results show that mTrader outperforms all other models on all metrics for the Bitcoin and Tesla datasets, achieving state-of-the-art results on the ASr and Tr metrics. The DQN model performs best on AVol and MDD, but its performance on Tr is much lower than the other models, so we guess that it may be trapped in a local optimal solution. Furthermore, we observed that the PPO algorithm outperforms other DRL algorithms, indicating that mTrader based on the PPO algorithm can generate higher excess returns, while other algorithms result in losses.

In the Bitcoin dataset, the overall trend was characterized by a rapid decline. As shown in Figure 3(a), mTrader not only avoided the peak of the decline, but also grasped the profit points in a short period of time when the whole market was in a downturn. This result demonstrates that the AO plays a crucial role in ensuring the correctness of the output action. In the Tesla dataset, mTrader’s ASr value far exceeds DQN, a method that has been proven to be extremely prominent in the financial field in other papers. Figure 3(b) shows that our method still avoided the most violent moment of fluctuation in the middle and obtained a profit more quickly than when it rose rapidly. In the CATL dataset illustrated in Figure 3(c), the overall fluctuation of our model was relatively stable, and the maximum return was reached at the last moment. The existence of AO allows the model to avoid market declines while limiting its performance in the rapid upward phase.

The above experiments can prove that our proposed model is capable of producing high returns while also being resistant to market downturns in various financial markets.

C. Impact of Hyper-Parameters

To further investigate the impact of feature selection on multi-scale signal modeling, we conducted experiments on the $m_1$ and $m_2$ parameters in AO. The results of these experiments, displayed in Figure 4, indicate that our proposed method achieved the best performance on all three datasets when $m_1 = 0.15$ and $m_2 = 0.01$. In contrast, performance decreased when $m_1 = 0.10$ and $m_2 = 0.005$. In two of the three experiments with $m_1 = 0.15$, the performance was higher than when $m_1 = 0.10$, suggesting that loosening the $P_2$ and $P_3$ boundaries of AO can enhance model performance.

Furthermore, the experiment on $m_2$ demonstrated that narrowing the boundary between $P_{h1}$ and $P_{h2}$ can lead to better trading outcomes. As $m_1$ and $m_2$ are kept within a reasonable range, our model outperforms the trading framework without AO, indicating that AO can effectively maximize returns and has sufficient generalization capabilities.

D. Ablation Experiments

We conducted ablation experiments on Bitcoin to evaluate the impact of different components of mTrader on its performance. In particular, mTrader-VQ refers to the method with the VQ-VAE module, and mTrader-AO refers to the method with the AO module. The results of these experiments are presented in Table III, and the cumulative return rates are summarized in Figure 5. The results suggest that the AO module significantly improves the stability of the method. Additionally, the method with the VQ-VAE module correctly responds to market upswings demonstrates the successful removal of noise and extraction of valid signals. From Figure 5, by leveraging the strengths of both modules, mTrader achieved a higher cumulative return rate compared to the other two methods for most of the time.

To further demonstrate the role of the VQ-VAE encoder, we use a VAE encoder [7] that produces a continuous distribution named mTrader-VAE. As shown in Table IV, each metric of mTrader-VAE is worse than using the VQ-VAE encoder. This means that the feature of discrete latent variables exploits the properties of our designed multi-scale signal matrix.
TABLE II: Performance of comparison methods. The best results are marked in **bold.**

<table>
<thead>
<tr>
<th>Models</th>
<th>Bitcoin</th>
<th>Tesla</th>
<th>CATL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ASr</td>
<td>Avol</td>
<td>MDD(%)</td>
</tr>
<tr>
<td>DQN</td>
<td>-0.37</td>
<td>0.59</td>
<td>24.27</td>
</tr>
<tr>
<td>DDPG</td>
<td>-1.01</td>
<td>0.64</td>
<td>22.16</td>
</tr>
<tr>
<td>PPO</td>
<td>0.01</td>
<td>0.35</td>
<td>11.49</td>
</tr>
<tr>
<td>Long &amp; Hold</td>
<td>-0.89</td>
<td>0.65</td>
<td>21.77</td>
</tr>
<tr>
<td>mTrader</td>
<td><strong>2.36</strong></td>
<td><strong>0.16</strong></td>
<td><strong>2.34</strong></td>
</tr>
</tbody>
</table>

TABLE III: Ablation experiments on Bitcoin.

<table>
<thead>
<tr>
<th>Models</th>
<th>Bitcoin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ASr</td>
</tr>
<tr>
<td>mTrader-AO</td>
<td>0.15</td>
</tr>
<tr>
<td>mTrader-VQ</td>
<td>-1.04</td>
</tr>
<tr>
<td>mTrader-VAE</td>
<td>-1.41</td>
</tr>
<tr>
<td>mTrader</td>
<td><strong>2.36</strong></td>
</tr>
</tbody>
</table>

VI. CONCLUSION

In this paper, we proposed a novel framework, mTrader, which uses multi-scale signal optimization deep reinforcement learning for financial trading. We introduced a complete feature augmentation and screening process to model minute-level data as multi-scale sparse matrices, and used VQ-VAE to generate discrete embeddings. We also incorporated the PPO and LSTM to effectively capture the time-series relationship. Our offline Action Optimizer (AO) further improved the stability of the trading framework. Our experiments on multiple markets with different properties confirmed the superiority of our proposed model.

We further aim to explore how to model macro financial indicators and the interplay between multiple stocks or cryptocurrencies, and even different markets. We also plan to test the trading framework on a real financial engine and explore the possibility of applying mTrader to other time-series tasks.

REFERENCES

Abstract—Aspect-Based Sentiment Analysis (ABSA) aims to determine the sentiment polarity of certain aspect words in a sentence. Recently, it is a popular approach to fuse the sentences’ syntactic information via the dependency tree into the graph neural network. However, how to efficiently utilize the obtained syntactic information is still a challenging problem of this kind of approach. Therefore, this paper proposes a novel Aspect-level Sentiment Analysis model based on Convolutional network with Dependency Tree, named ASAC-DT in short. First, the attention mechanism is utilized to obtain the attention score of the sentence and the aspect word respectively, to improve the connection of the words related to the aspect word in the sentence. Afterwards, by relying on the syntactic information obtained from the dependency tree, the connections of words that are not related to the aspect words are reduced. Finally, the feature information most relevant to the aspect words in the proposed model is extracted through the graph convolutional neural network and the interactive network. Through extensive experimental baselines the proposed ASAC-DT model shows effectiveness in aspect-level sentiment classification and outperforms baselines in accuracy.

Keywords-Aspect-Based Sentiment Analysis; Attention mechanism; GCN; Interactive network

I. INTRODUCTION

In sentiment analysis research on texts, aspect based sentiment analysis (ABSA) aims to analyze the sentiment polarity of related aspect terms in sentences. The analysis of the emotional polarity of different aspect terms in a sentence is defined as the aspect-level sentiment analysis (ASA), and has been a focus of textual sentiment analysis in recent years.

ASA is constantly evolving with deep learning techniques, and the techniques used are constantly being updated and iterated. One of the key points is to establish a link between aspect and viewpoint words. Among the studies in recent years that have modeled the link between aspect and viewpoint words. The AT-LSTM [1] and RAM [2] models emphasize the use of AM to model the relevance between aspect words and contexts. However, because of the complexity of textual language, the use of the attention mechanism may lead to compromised connections between aspect and viewpoint words.

Due to the specificity of graphical neural networks, the study of syntactic information in combination with sentences has also become a trend in party-level sentiment analysis. The information on the syntactic structure of the sentence enables a better strengthening of the connection between aspect and viewpoint words. For example, the ASGCN [3] model extracts the syntactic information of a sentence after processing it and then extracts the sentence feature information via GNN [4]. The CDT [5] model combines syntactic information with word embeddings by acquiring sentence feature information by the use of a graph convolutional network (GCN) to enhance the presentation of the learned aspect words. However, in both studies, neighboring nodes were given the same weight, without distinguishing between the different importance of neighboring nodes to the current node. Therefore, the noise information in it will have a certain impact on the accuracy of the model.

In addition, many ASA studies based on GNNs incorporate syntactic information about sentences, but don’t sufficiently utilize syntactic information. In some complex sentence structures, aspect and viewpoint nodes are linked to each other by different dependencies. In the process, irrelevant node information may be introduced, creating a noisy impact, which is a challenge in the current research.

Therefore, we propose a novel Aspect-level Sentiment Analysis model based on Convolutional network with Dependency Tree, named ASAC-DT in short. First, the sentence representation is obtained by Bi-LSTM to obtain a hidden layer representation of the sentence. Afterwards, the attention mechanism (AM) was utilized to calculate the self-attention scores and the aspect words respectively. The dependency tree generated by the sentence representation is then constructed into a graph-related representation. Both the attention mechanism and the dependency tree representation of sentences can strengthen the links between aspect and viewpoint words in a sentence, while reducing the interference of irrelevant words. Finally, the attention score and sentence dependency tree representation are integrated through the GCN network to maximize the use of effective information in the sentence.

In summary, the main contributions of the proposed model are as follows:

- Construct and use sub-trees of the dependency tree with attention scores, to reduce the effect of noise.
- Applying attention scores directly to the GCN network with the interaction network to ensure maximum extraction of sentence feature information.

DOI reference number: 10.18293/SEKE2023-127
• Experimental results demonstrate that the proposed model achieves better results than the baseline method on the public datasets.

II. RELATED WORK

A. Aspect-level sentiment analysis

With the continuous development of the research process, in the early related research [6], the task of predicting emotional polarity in the ABSA was realized by manually defining the relevant syntactic rules. In addition, some studies [7-9] combined the relevant neural network in deep learning to improve the efficient of the sentence feature information extraction. Among them, Li [7] effectively utilizes aspect words and context words to provide information utilization of the context of the sentence, but does not effectively consider other structural information of the sentence. Wang [8] proposed a hierarchical network that focuses on both the word and the clause level. However, it ignores the sentence text information. Fan [9] proposed a new multi-granularity attention model, but it cannot solve the noise problem generated in the model well.

B. Syntax Analysis

Applying syntactic information to sentence representation is a commonly used research mode recently. The adaptive recurrent neural network (AdaRNN) [10] transferred the emotional tendency to the corresponding target word according to the context and syntactic structure information. However, it is easily affected by noise data during data transmission. Nguyen [11] optimized and expanded AdaRNN, combining the dependency tree and composition tree of the sentence into the structural information of the sentence. The CDT [12] model directly integrates the syntactic information of the dependency tree into the word embedding information. The DSS-GCN [13] model combined syntactic information, semantic information, and structural information of sentences to improve the connection between words in sentences.

Inspired by the CDT model, the syntactic information of the dependency tree is adopted in the proposed ASAC-DT model. Different from the CDT, the syntactic information of ASAC-DT is split according to the corresponding dependency tree structure information. Also ASAC-DT constructed the subtree of the dependency tree, so as to capture the syntactic information much better.

C. GCN

Many ASA researches have begun to revolve around GCN with the AM. The relational graph attention (R-GAT) model[14] has achieved good results in prediction through the effective use of AM. However, sentence information is affected by dependencies during the encoding. Chen [15] proposed a model that integrates GCN and common AM, and removes the effect of noise in contextual words, but does not use the dependence on syntactic information.

A model [3] applied GCN on the dependency tree of a sentence to effectively capture syntactic information and remote word dependencies. However, the model does not make sufficient use of the syntactic information captured. The HL-GCN model [16] improved the dependency between sentence representations by dividing different dependency subtrees. The CANN-SSCG model [17] integrated the sentiment knowledge, syntactic information and contextual information of the sentence, followed by a GNN to extract the effective information of the sentence.

The proposed ASAC-DT model incorporates syntactic information and AM into the GCN together to improve the information extraction capability of the GCN as well as to reduce the interference of noisy data.

III. METHODOLOGY

As shown in Fig.1, the ASAC-DT model consists of six parts, each of which has the following roles.

• Input and encoding layer. The word embedding technique obtains a vector representation of the sentence.
• Attention layers. The AM was used to calculate the self-attention scores of the sentences and the attention scores about the aspect words separately.
• Dependency tree layer. The syntax parser is used to obtain information about the dependency tree structure of the sentence.
• Graph convolution layer. The ASAC-DT model integrates the hidden layer information of the sentence, the sentence dependency tree structure information and the attention score for feature extraction. Interaction network layer. Fusion extracts useful information from sentences and reduces the loss of data information during convolution.
• Output layer. Output the final classification result of the model.

A. Input and Encoding Layer

The sentence representation \( s = \{w_1, w_2, ..., a_1, a_m, ..., w_n\} \) is obtained by the word embedding technique, where \( \{a_1, ..., a_m\} \) is the aspect word representation of the sentence. Afterwards the data is entered into the encoding area.

Bi-LSTM has been utilized here to extract sentence feature information. The word embedding representation of a sentence is obtained by the GloVe technique with the word embedding representation \( x = \{x_1, ..., x_{a+1}, x_{a+m}, ..., x_n\} \). Afterwards the Bi-LSTM extracts the feature information of the sentence to obtain the hidden state vector \( H = \{h_1, h_2, ..., h_{a+1}, ..., h_{a+m}, ..., h_n\} \) of the sentence, where \( h_t \in \mathbb{R}^{2d} \) denotes the hidden state at time step \( t \). \( d \) denotes the dimension of the hidden state vector in the one-way LSTM.

The word embedding representation \( x \) is simply represented by the following equations (1-3).
\[
\overline{h_t} = LSTM(x_t) \quad (1)
\]
\[
\overline{h_t} = LSTM(x_t) \quad (2)
\]
\[
h_t = \left[ \overline{h_t}, \overline{h_t} \right] \quad (3)
\]

Where:
- \( x_t \) indicates word embedding representation.
- \( h_t \) defines the hidden state at the time \( t \).

### B. Attention Layers

The ASAC-DT model calculates the self-attention scores of the sentences and the attention scores about the aspect words with the AM[18]. The formula is as follows (4):

\[
Attention(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_w}} \right) V \quad (4)
\]

Where:
- Given a key \( K \), a query \( Q \) and a value \( V \).
- \( d_w \) indicates the dimension of the word representation in the sentence representation.

1) Calculation of self-concentration score: The sentence hidden state vector \( H \) is obtained from the input and encoding layers and then assigned to \( K, Q \) and \( V \) respectively. The self-attention score of the sentence is obtained by calculating the Attention\( (Q, K, V) \) formula shown as (5).

\[
\text{self}_{\text{att}} = Attention(H, H, H) = \text{softmax} \left( \frac{HH^T}{\sqrt{d_w}} \right) H \quad (5)
\]

2) Calculation of aspect word attention score: The hidden state \( H_a \) of the aspect word is obtained from the hidden state representation in the following (6).

\[
H_a = \text{mask}(H) \quad (6)
\]

The mask function obtains the aspect-word representation \( H_a = [0,0, ..., h_{a+1}, ..., h_{a+m}, ..., 0] \) by setting 0. Assign \( H_a \) to \( Q \) and \( H \) to \( K \) and \( V \) to obtain the aspect-word representation's attention score aspect-attention calculated as (7).

\[
\text{asp}_{\text{att}} = Attention(H_a, H, H) = \text{softmax} \left( \frac{H_aH^T}{\sqrt{d_w}} \right) H \quad (7)
\]

The attention layer acquires the sentence's self-attention score self-attention and the aspect word's attention score aspect-attention.

### C. Dependency Tree Layer

The sentences form a dependency tree structure through the parser. This dependency structure preserves the dependencies between each word in the sentence. This is shown in Fig. 2 below.

A sentence dependency tree is transformed into a graphical representation of the dependency structure of the sentence \( d^{\text{sym}} \). The dependency graph representation of the sentence \( d^{\text{sym}} \) is partitioned into three sub-tree dependency structure graph representations \( \{d_1^{\text{sym}}, d_2^{\text{sym}}, d_3^{\text{sym}}\} \) according to the syntactic distance \( \{1, 2, 3\} \).

### D. Graph Convolution Layer

In the proposed ASAC-DT model, a GCN is utilized to process on further extraction of sentence features. The calculation formula is described as the following (8):

\[
H^{\text{out}} = GCN(H, D) \quad (8)
\]

Where:
- \( H \) is represented as a sentence vector.
- \( D \) is represented as a sentence adjacency matrix.

![Figure 2. Diagram of the sentence dependency structure](image-url)
Here the hidden state vector $H$ generated by the encoding layer, the self-attentive score $self_{att}$ of the sentence generated by the attention layer are used as the input of the GCN. The self-attentive scores assign weights based on the weight relationships of the words in the sentence, making it more efficient to extract information about the sentence features. The final output is $H^{att}$.

At the same time, this model integrates the GCN to make it more suitable for the information extraction ability of the model. As shown in Fig. 3. The integrated calculation formula is as follows (9-10):

$$
H^d = GCN(H, D^{sym})
$$

(9)

$$
A_d = H^d \otimes att
$$

(10)

Where:

- $H$ is represented as a sentence vector.
- $D^{sym}$ is represented as a sentence dependency graph.
- $att$ represents the attention score.

Calculated according to the self-attention score $self_{att}$ of the sentence and the aspect word attention score $asp_{att}$. First, the hidden state representation $H$ of the sentence in the encoding layer and the dependency structure graph representation $D^{sym}$ of the sentence in the dependency tree layer are used as the input of $GCN(H, D^{sym})$, and then the self-attention score of the sentence obtained by the attention layer $self_{att}$ to operate. Get an output result $A_d \in \mathbb{R}^{2d}$.

At the same time, the hidden state representation $H$ of the sentence in the encoding layer and the three subtree dependency structure graphs of the sentence represent $\{d_1^{sym}, d_2^{sym}, d_3^{sym}\}$. As the input of $GCN(H, D^{sym})$, it is operated with the aspect word attention score $asp_{att}$ of the sentence obtained by the attention layer respectively, and three output results $\{A_{d1}, A_{d2}, A_{d3}\} \in \mathbb{R}^{2d}$.

**E. Interaction Network Layer**

Before the graph convolution layer data is transmitted, an average pooling operation will be performed. The calculation method is as follows (11).

$$
f^{out} = \text{Average pooling}(f)
$$

(11)

The output results $H^{att}$, $A_d$ and $\{A_{d1}, A_{d2}, A_{d3}\}$ obtained by the graph convolution layer are subjected to an average pooling operation. Before pooling, the integration operation is performed on $\{A_{d1}, A_{d2}, A_{d3}\}$. It is calculated as follows (12).

$$
A_{asp} = [A_{d1} \oplus A_{d2} \oplus A_{d3}]
$$

(12)

Perform interactive processing on the pooled results $f^{att}$, $f^{Ad}$ and $f^{A_{asp}}$ of $H^{att}$, $A_d$ and $A_{asp}$. This paper adopts a simple and effective cross-network structure [19]. The calculation formula of interaction is as follows (13).

$$
X_{t+1} = X_0 X_1^T W_t + b_t + X_t
$$

(13)

Where:

- $X_t \in \mathbb{R}^d$ is the input of the interaction network.
- $X_{t+1} \in \mathbb{R}^d$ is the output.
- $W_t \in \mathbb{R}^d$ is the weight parameter.
- $b_t \in \mathbb{R}^d$ is a bias item.

The calculation process of the simplified interaction in this paper is as follows (14-15):

$$
X_A^{dc} = f^{H_{att}} (f^{Ad})^T + f^{Ad}
$$

(14)

$$
X_A^{asp} = f^{Asp} (f^{Ad})^T + f^{Ad}
$$

(15)

Where:

- $X_A^{dc} \in \mathbb{R}^a$ and $X_A^{asp} \in \mathbb{R}^a$ are the output results after interaction.

**F. Output Layer**

In the output layer, the two obtained outputs $X_A^{dc} \in \mathbb{R}^a$ and $X_A^{asp} \in \mathbb{R}^a$ are spliced first, and then sent to the classifier. The output process is shown in formula (16-17).

$$
o = [X_A^{dc}, X_A^{asp}]
$$

(16)

$$
o' = \text{softmax}(\text{Linear}(o))
$$

(17)

Where:

- $o \in \mathbb{R}^{2a}$ is the spliced output.
- $o' \in \mathbb{R}^{out}$ is the final output. $out$ is the number of classifications of the final output.

The loss function of the ASAC-DT model is the cross-entropy loss function (CrossEntropy Loss), the specific formula (18) is as follows:

$$
H(p, q) = - \sum_x p(x) \log q(x) + \lambda ||\theta||^2
$$

(18)

Where $p(x)$ is the true distribution of the sample, and $q(x)$ is the predicted distribution of the sample. At the same time, $L_2$ regularization is also added for constraints. $L_2$ regularization controls the complexity of the model and reduces the over-fitting of the model.

**IV. Experiment Result and Analysis**

**A. Datasets**

For experimental evaluation, this paper uses three public datasets, Lap14 and Rest14 in Twitter datasets [10] and
the model. It also outperforms models using dependency trees and GCNs such as CTD, BiGCN and kumaGCN in terms of effectiveness in capturing aspect and viewpoint words.

E. Number of Att-GCN Layer

In this paper, we conducted a comparative experiment on the effect of the number of layers of Att-GCN on the accuracy of the model, and the results are shown in Fig. 4.

As can be seen in Fig. 4, the number of layers set up for the experiment was 3, 4, 5, 6 and 7, with varying degrees of decrease in the accuracy of the model as the number of layers increased. It shows that an increase in the number of layers can lead to overfitting of the model.

F. Ablation Experiment

The corresponding ablation experiments were also performed in this paper, as shown in Table IV.

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dropout_rate</td>
<td>Word embedding layer</td>
<td>0.7</td>
</tr>
<tr>
<td>batch_size</td>
<td>Size of mini-batch</td>
<td>16</td>
</tr>
<tr>
<td>r</td>
<td>Initial learning rate</td>
<td>0.001</td>
</tr>
<tr>
<td>d_e</td>
<td>Size of embedding layer</td>
<td>300</td>
</tr>
<tr>
<td>d_h</td>
<td>Size of hidden layer</td>
<td>300</td>
</tr>
<tr>
<td>l2</td>
<td>L2-Regularization weight</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

The proposed ASAC-DT model has been compared with the following baselines.

- **AS-GCN[3]:** Applying GCN to sentiment analysis to extract feature information of sentences efficiently.
- **CDT[5]:** The dependency tree and CNN are modeled to learn sentence feature information.
- **BiGCN[23]:** Hierarchical modeling is carried out through the syntactic structure and lexical information of the sentence itself.
- **kumaGCN[24]:** Apply dependency graphs and latent graphs to models to improve the model performance.
- **DGEDT[25]:** Through the flat representation of the transformer, a dual-transformer model augmented by a dependency graph is designed.

D. Results Analysis of Comparison Experiments

The comparison results of the ASAC-DT model and other models are shown in Table III. As can be seen from the data in the table, the model in this paper outperforms most of the baseline methods in terms of accuracy and Macro-F1. It is shown that the attention mechanism used by the text model and the integration of the dependency tree structure information into the GCN has led to some improvement in the performance of

<table>
<thead>
<tr>
<th>Method</th>
<th>Twitter</th>
<th>Lap14</th>
<th>Rest14</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AS-GCN</strong></td>
<td>72.15</td>
<td>70.40</td>
<td>75.55</td>
</tr>
<tr>
<td><strong>CDT</strong></td>
<td>74.66</td>
<td>73.66</td>
<td>77.19</td>
</tr>
<tr>
<td><strong>BiGCN</strong></td>
<td>74.16</td>
<td>73.35</td>
<td>74.59</td>
</tr>
<tr>
<td>kumaGCN</td>
<td>72.45</td>
<td>70.77</td>
<td>76.12</td>
</tr>
<tr>
<td>DGEDT</td>
<td>74.80</td>
<td>73.40</td>
<td>76.80</td>
</tr>
<tr>
<td><strong>ASAC-DT</strong></td>
<td>76.16</td>
<td>74.47</td>
<td>77.59</td>
</tr>
</tbody>
</table>

TABLE III. RESULTS OF COMPARISON EXPERIMENTS

<table>
<thead>
<tr>
<th>Method</th>
<th>Twitter</th>
<th>Lap14</th>
<th>Rest14</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AS-GCN</strong></td>
<td>72.15</td>
<td>70.40</td>
<td>75.55</td>
</tr>
<tr>
<td><strong>CDT</strong></td>
<td>74.66</td>
<td>73.66</td>
<td>77.19</td>
</tr>
<tr>
<td><strong>BiGCN</strong></td>
<td>74.16</td>
<td>73.35</td>
<td>74.59</td>
</tr>
<tr>
<td>kumaGCN</td>
<td>72.45</td>
<td>70.77</td>
<td>76.12</td>
</tr>
<tr>
<td>DGEDT</td>
<td>74.80</td>
<td>73.40</td>
<td>76.80</td>
</tr>
<tr>
<td><strong>ASAC-DT</strong></td>
<td>76.16</td>
<td>74.47</td>
<td>77.59</td>
</tr>
</tbody>
</table>

The comparison results of the ASAC-DT model and other models are shown in Table III. As can be seen from the data in the table, the model in this paper outperforms most of the baseline methods in terms of accuracy and Macro-F1. It is shown that the attention mechanism used by the text model and the integration of the dependency tree structure information into the GCN has led to some improvement in the performance of
Removing the aspect-level attention score (w/o aspect-attention) from the model reduced the accuracy of the model by 0.73%, 1.11% and 0.36% in the three datasets Twitter, Lap14 and Rest14. At the same time, the corresponding dependent subtree structure (w/o dependency subtree) in the model is removed, and the accuracy decreases by 2.47%, 1.58% and 3.13%.

Because the aspect-attention is in the Att-GCN between the integration and the dependent subtree, when the dependent subtree structure is removed, the corresponding aspect-attention will also be removed. From the results of the data, it can be seen that aspect-attention plays an important role in the model and strengthens the correlation with aspect words.

Relying on the subtree structure, the influence of noise data is also reduced in the acquisition of syntactic information. The corresponding accuracy of the ablation of the interaction network (w/o cross network) on the three data sets decreased by 1.31%, 1.42% and 0.63, respectively. It shows that the interactive network also plays a certain role in the integration of sentence information.

V. CONCLUSION

The proposed ASAC-DT model is a GCN model based on attention mechanism and dependency tree. The interaction network is used in the model for aspect-level sentiment analysis tasks. According to the particularity of the graph convolutional neural network, the corresponding attention mechanism score and the syntactic dependency information generated by the dependency tree are integrated into it. The dependency relationship obtained through the attention mechanism and the syntactic information provided by the dependency tree enable the model to more accurately capture the connection between aspect words and opinion words. Among them, the semantic information and syntactic information of the sentence have been effectively transmitted, which makes the model have a more effective text sentiment classification ability. The experimental results show that the experimental model in this paper is superior to the current baseline model and has achieved good results on various public datasets.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (61902116).

REFERENCES

Abstract

Monitoring the functionality of systems during operation is vital for detecting faults and preventing their consequences. In autonomous driving, monitoring is even more critical because of hardly being able to verify all implemented functionality. Today, systems comprise many interacting components making centralized monitoring less feasible and hard to handle. Hence, we suggest a distributed but connected monitoring system that reflects the system’s conceptual structure. In this paper, we outline the foundations of a monitoring system, present some applications and show how we use concepts like the operational design domain and requirements for obtaining the required monitoring knowledge in the application area of autonomous driving.

1 Introduction

Autonomous driving has gained much attention in the past years. Driving factors are reduced costs, the opening of new opportunities for services, and the reduction of fatalities, among others. When focusing on decreasing fatalities in driving, we have the underlying assumption that autonomous driving is safe or at least safer than human driving. Checking that a concrete implementation of autonomous driving reaches this goal requires sophisticated verification and validation before deployment. Many researchers have been working on this topic, including Koopman and Wagner [6], Wotawa [11], Schuldt and colleagues [9], or Wotawa and colleagues [13].

The underlying challenge is to find a way of justifying the quality of verification and validation. For this purpose, Kalra and Paddock [5] stated that an autonomous vehicle has to operate for 275 million miles for verification purposes. In their calculation, the authors took the fatality rate of driving in the USA and assumed that autonomous vehicles should be far safer. Unfortunately, testing for 275 million miles is not feasible in practice. Therefore, other researchers have proposed using ontologies for testing automated, autonomous vehicles, e.g., [14, 2, 7]. These contributions focus on testing potential unsafe interactions between the autonomous car and its environment. Note that there is a massive number of potential interactions between a car and its environment, indicating the need for additional means for verifying the behavior of autonomous vehicles.

One way to reduce the search space for verification and validation is to restrict the use domain. And indeed future electric connected and automated (ECA) vehicles will be deployed in specific targeted operational design domains (ODD) across Europe [4], [8] able to safely and robust operate in the target ODD making use of their embedded behavior competences [10]. The recently published and continuously amended EU ADS regulation [1] outlines some general safety regulations L4+ vehicles to be deployed within Europe have to comply with. Especially the outlined pillar III, referred to as in-service monitoring, highlights the importance of monitoring activities during the operation of future ECA vehicles after they enter the market and get deployed on public roads. In-service monitoring and reporting targets to learn from in-service data as a central component to the safety potential of future ECA vehicles enabling key objectives referred to safety confirmation, scenario generation, and safety recommendation representing a major source of general safety requirements of future ECA vehicles.

Figure 1 outlines the general architecture of a monitoring system. First, the monitoring system receives internal values from the system under supervision using a monitoring interface. The internal values comprise sensor inputs, internal states, error signals, and actuator commands over time. Next, these values are checked for consistency with expectations using a knowledge base. In case of inconsistencies, the monitoring system raises an error or warning,
Figure 1: The proposed monitoring architecture.

depending on the severity, which the mitigation mechanism uses for suggesting corrective or compensating actions. Finally, mitigation transfers these actions to the system using the mitigation interface. It is worth stating that the monitoring system is independent of the system under supervision and only communicates via the proposed interfaces. The system itself communicates with its environment using actuators and sensors.

Because of the hierarchical nature of systems comprising subsystems down to basic components (at least from a conceptual point of view), Lewitschnig and Wotawa introduced a hierarchical monitoring concept. In this paper, we extend this concept to a general monitoring paradigm comprising separate monitoring nodes that are interconnected and which belong to parts of a system. Each monitoring node checks the behavior of its corresponding parts, also considering results from other monitoring nodes. The monitoring nodes perform the checks utilizing formalized knowledge of the expected behavior. In case of severe violations, we assume the existence of mitigation actions like safety maneuvers. However, we do not discuss how mitigation will be carried out in this paper and leave this for future research. In addition, to knowledge-based monitoring, we further discuss how to obtain knowledge by considering ODDs and hierarchically structured monitoring systems. Finally, we illustrate the process using examples from the automotive industry.

We structure this article as follows. We start motivating research in knowledge-based monitoring using the case of an accident of a Tesla car driving in autonomous mode (Sect. 2). Afterward, Section 3 introduces the foundations of monitoring and its specialization into hierarchical monitoring devices. Then, in Section 4, we introduce a process for obtaining the required knowledge using ODD and illustrate its use. Finally, we conclude the paper.

2 Motivation

In this section, we motivate the need for monitoring autonomous driving, considering different views. First, let us discuss an accident of a Tesla car driving in autonomous mode on a highway in Taiwan in 2020, where we depict the most important parts of the accident’s chronology in Figure 2. The Tesla car was not reducing speed until braking came too late. As a result, the Tesla crashed into an overturned truck that blocked the two left lanes of the highway. It seems that Tesla’s vision systems did not perceive the truck. Hence, there might be an issue with the vision system that an autonomous car can hardly identify without perception redundancies. However, even worst, the Tesla was not even reacting to the truck driver on the highway, who warned drivers. Such behavior does not belong to any road traffic regulation.

The Convention on Road Traffic emerged from the United Nations Conference on Road and Motor Transport in September 1949 in Geneva. In the document Article, 7 and 10 are of particular interest to us. Article 7 states: “Every driver, pedestrian or other road user shall conduct himself in such a way as not to endanger or obstruct traffic: he shall avoid all behaviour that might cause damage to persons, or public or private property.”. Especially the last part is of interest to us when considering the Tesla accident, where the car is passing by the truck driver at high speed, not considering potential damage to persons. This behavior also conflicts with Article 10: “The driver of a vehicle shall at all times have its speed under control and shall drive in a reasonable and prudent manner. He shall slow down or stop whenever circumstances so require, and particularly when visibility is not good.”. Of course, in the mentioned case, the Tesla driver is responsible for violating the traffic convention. However, the car in autonomous driving mode was not implementing the regulations too.

We can derive from the Tesla accident example: There is a need for thoroughly verifying the vision system, considering different and even unexpected scenarios. In order to detect faulty behavior of an autonomous car while driving, it is essential to have redundancies in perception. Otherwise, we are not able to detect perception faults. In addition, we need to formalize traffic rules and check them during operation. For example, in the Tesla accident, the control system might have ignored the truck driver due to other regulations preventing persons from walking on highways. An independent monitoring system relying on formalized traffic regulations would detect such a fault and be able to initiate countermeasures like braking or changing lanes. In any case, a monitoring system utilizes perceived information.
3 Monitoring Foundations

In the following, we outline the foundations behind knowledge-based monitoring. We extend previous work \cite{12} to handle monitoring systems of all kinds of structures, not just hierarchical ones, i.e., tree-structured monitoring systems.

**Definition 1 (Monitoring System)** A tuple $(COMP, CONN)$ is a monitoring system where $COMP$ is a set of components, and $CONN$ is a set of connections between components, i.e., a set of pairs from $COMP \times COMP$.

Because of simplicity, we further introduce the functions for obtaining the predecessor and successor of a component $C \in COMP$, i.e., $\text{pred}(C) = \{C'|(C', C) \in CONN\}$, and $\text{succ}(C) = \{C'|(C, C') \in CONN\}$ respectively, and the definition of basic nodes, i.e., nodes that do not have successors, $\text{basic} = \{C|(C', C) \in CONN\} \land \text{succ}(C) = \emptyset$.

What we need to add is the monitoring and checking of the behavior of a system. For this purpose, we introduce three additional functions we must define for each monitoring system component. We assume, without restricting generality, to use first-order logic (FOL) for representing the information delivered by these functions.

**Output:** The $\text{out} : COMP \rightarrow FOL$ function delivers a set of predicates to hold when executing the given component.

**Constraints:** The $\text{constr} : COMP \rightarrow FOL$ function delivers first-order logic sentences that must hold in the context of a given component.

**Behavior:** The $\text{behav} : COMP \rightarrow FOL$ function returns a set of first-order logic sentences that are used with the inputs of a component (i.e., the outputs of the successor nodes) to derive the outputs.

We use these functions to express the behavior of the system. For each component $C_i$, we can derive its output from...
Algorithm 1 monitoring\((O, (COMP, CONN))\)

**Input:** A set of observations \(O\) obtained from the system under the supervision and a monitoring system.

**Output:** \(\top\) if all constraints are fulfilled, and \(\bot\) otherwise

1: Convert each observation \(o \in O\) to its corresponding predicate \(p\) and assign it to a corresponding monitoring system component \(C \in COMP\).
2: while There is no change in \(out(C)\) of any \(C \in COMP\) do
   3:   for all \(C \in COMP\) do
      4:      for all Fact \(P\) that can be derived from \(behav(C)\) ∪ \((\bigcup_{C' \in \text{pred}(C)} out(C'))\) do
         5:         Add \(P\) to \(out(C)\).
      6:   end for
      7:   if \(\text{constr}(C) \cup out(C) \neq \bot\) then
         8:         Add violation() to \(out(C)\)
      9:   end if
   10: end for
11: end while
12: if \(\exists C \in COMP : \text{violation()} \in out(C)\) then
   13:   return \(\bot\)
15: else
   16:   return \(\top\)
   end if

the inputs considering the defined behavior of the component. Moreover, if each component \(C_i\) works correctly, all constraints must be fulfilled considering given inputs and computed output. We use both rules in the monitoring algorithm (Algorithm 1), called every predefined time step \(t\). In the first line of the algorithm, there is a mapping of current observations at time \(t\) to corresponding predicates of the monitoring system components. Afterward, we apply derivations until no new facts can be derived anymore. Next, the algorithm derives new facts in lines 4–6, where we use the inputs, i.e., the outputs of predecessors of the current component, and the given behavior of the component to derive new facts \(P\). Afterward, the algorithm checks the fulfillment of given constraints using the current facts in Line 7.

Algorithm monitoring terminates because, in each step, it adds new facts to \(out(C)\) for the current component \(C\). Adding facts is done until no new facts can be derived. Because we consider only a finite number of facts, the algorithm must terminate. It is also worth mentioning that the algorithm transfers all facts of any node to its successor. If this is not wanted, we need to add a filtering rule. Further note that the algorithm generalizes a previous one [12].

We illustrate the monitoring foundations using a small example where we formalize a monitoring system that would have at least indicated a wrong driving behavior for the Tesla accident from Figure[2]. We assume three monitoring components: \(C_{gps}\) for gaining information about speed and the location, i.e., whether the car is on a highway or not accordingly to given maps, \(C_{vision}\) for indicating other vehicles or persons, and \(C_{car}\) for the overall Tesla car. Obviously, \(C_{gps}\) and \(C_{vision}\) are predecessors of \(C_{car}\). Hence, we represent the monitoring system using the tuple \(\{\{C_{gps}, C_{vision}, C_{car}\}, \{C_{gps}, C_{car}\}, (C_{vision}, C_{car})\}\).

For all components, we do not add behavior in this example. For \(C_{gps}\), we assume that predicates \(\text{onHighway()}\), \(\text{speed}(v)\), and \(\text{braking()}\) indicate the operation on the highway, the current speed \(v\) and information regarding braking, respectively. Note that the predicates can also be available in their negated form using the operator \(\neg\). For \(C_{vision}\), we have a predicate \(\text{personOnLane()}\) indicate whether a person is on the driving lane within reach. For \(C_{car}\), we do not introduce other predicates. Furthermore, for \(C_{gps}\), we add a constraint \(\neg(\text{onHighway()} \land \text{speed}(v) \land v > 130)\) stating that overspeeding on the highway is not allowed considering the speed limit in Austria, which is \(130\)km/h. For component \(C_{car}\), we assume that we have to brake once a person is on the lane within reach, i.e., \(\neg(\text{personOnLane()} \land \neg\text{braking()}\)\).

Let us now apply the monitoring system to the Tesla accident, considering the third time step. In this case \(\text{onHighway()}\), \(\text{personOnLane()}\), and \(\neg\text{braking()}\) must be true. We assume that the speed is less than \(130\)km/h, e.g., \(100\)km/h, leading to the predicate \(\text{speed}(100)\). After mapping all observations to predicates, we obtain the following set of predicates for each component:

<table>
<thead>
<tr>
<th>(C)</th>
<th>(out(C))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_{gps})</td>
<td>(\text{onHighway(), speed}(100), \neg\text{braking()})</td>
</tr>
<tr>
<td>(C_{vision})</td>
<td>(\text{personOnLane()})</td>
</tr>
<tr>
<td>(C_{car})</td>
<td>(\text{onHighway(), speed}(100), \neg\text{braking()})</td>
</tr>
</tbody>
</table>

After applying algorithm monitoring, all predicates are also available in \(C_{car}\) because of lines 4–6.

<table>
<thead>
<tr>
<th>(C)</th>
<th>(out(C))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_{gps})</td>
<td>(\text{onHighway(), speed}(100), \neg\text{braking()})</td>
</tr>
<tr>
<td>(C_{vision})</td>
<td>(\text{personOnLane()})</td>
</tr>
<tr>
<td>(C_{car})</td>
<td>(\text{onHighway(), speed}(100), \neg\text{braking()})</td>
</tr>
</tbody>
</table>

\(\text{personOnLane()}\) and \(\neg\text{braking()}\) violate the constraint \(\neg(\text{personOnLane()} \land \neg\text{braking()}\) of \(C_{car}\). When considering that mitigation would stop the car because of the constraint violation, the accident would have been prevented.
4 Autonomous driving use case

To maximize the learning from in-service data, we developed a hierarchical monitoring device concept including 4 defined ECA vehicle layers (starting from subcomponents, via components, through sub-systems up to the system, the entire ECA vehicle itself). The main purpose of the hierarchical monitoring device approach is to monitor the health status of the individual elements across the 4 specified layers to guarantee the safe operation of the vehicle within its specified ODD. The nominal behavior of the 4 specified layers is directly specified via the system requirements and specification of the entire ECA vehicle. Compare requirements flow is highlighted in blue in Figure 3. The ECA vehicle requirements originate from the target ODD and related behavior competencies needed to safely operate within the target ODD completed by certification, homologation, legislation, and standardization of relevant aspects. Out of those 4 requirements sources, the system requirements are specified and pushed down along the 4 ECA vehicle layers (ECS value chain).

Contrary to the requirements flow (down-down from the system level), the health status flow moves bottom-up, see again Figure 3. Each layer determines the health status of its included elements and communicates them up to the next layer until the system layer (ECA vehicle) is reached.

All monitoring activities within the specific layers are operating according to the same receive-monitor-transmit principle. The receiving element collects all information about the individual health status of the current and previous layer to have a complete database ready which forms the basis for the subsequent monitoring stage. In detail, the monitoring element classifies the health status of the current layer based on the collected data of the receiving element analyzing the current residual risk for continuation. In case some misbehaviors or failures are observed with a certain layer the following strategy will be applied:

- Handle, correct, and mitigate all occurring failures and misbehavior if possible within the current layer making use of e.g. redundant elements, etc.

- In case a correction is not possible within the current layer, classify the health status (optimal, acceptable, critical) based on the available information to be transmitted to the next layer.

Finally, the transmitting element transfers the classified health status of the current layer to the subsequent one. In case the system layer is not able to correct eventually occurring faults and misbehavior directly, the ECA vehicle has to reduce its specified manageable target ODD and/or behavior competencies to further guarantee a safe operation and avoid unacceptable residual risks. In other words, the major goal behind that approach is to continue the operation as long as the residual risk in reduced ODD and behavior competencies is acceptable again as a safe-stop is not always the best option in case of faults and observed misbehavior. The reduction of the target ODD or behavior competencies can be applied several times until the mandatory common basis gets lost, e.g. on the highway the ECA vehicle is just able to drive 30km/h anymore. This operation is against the traffic law which makes a continuation impossible. In that aspect, a so-called minimum risk maneuver has to be trigger that safely terminates the operation of the ECA vehicle e.g. on the emergency lane of the highway.

5 Conclusions

This paper discussed the importance of monitoring focusing on autonomous driving and introduced a general monitoring framework comprising interconnected monitoring nodes. Besides the formal foundations, we outline its use considering a small real-world example. In addition, we discuss how monitoring knowledge can be obtained using the operational design domains and how to handle faults in the monitoring framework. Future work will include implementing the proposed approach in an autonomous driving demonstrator and conceiving an experimental evaluation.

Acknowledgement

ArchitectECA2030 receives funding within the Electronic Components and Systems For European Leadership Joint Undertaking (ESCEL JU) in collaboration with the European Union’s Horizon2020 Framework Programme and National Authorities, under grant agreement n° 877539. All ArchitectECA2030 related communication reflects only the author’s view and the Agency and the Commission are not responsible for any use that may be made of the information it contains. The work was partially funded by the Austrian Federal Ministry of Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) under the program "ICT of the Future" project 877587.

References


Figure 3: Requirements vs. health status flow.


Fake News Detection with Context Awareness of the Publisher

Xingchen Ding, Chong Teng*, Donghong Ji
Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education,
School of Cyber Science and Engineering, Wuhan University, Wuhan, China
Email: {xingos, tengchong, dhji}@whu.edu.cn

Abstract—The spread of fake news is a significant social problem that can have disastrous impacts on various domains, such as politics and the economy. Therefore, detecting fake news has become a major concern. However, prior research has relied solely on news text to derive news representation, which is inadequate because different news items under the same publisher are interconnected. To address this limitation, we propose an innovative approach called the Publisher-oriented Multi-view Graph Model (PMGM) that leverages the context awareness of the publisher to detect fake news. Our approach enriches the news representation by incorporating publisher profiles and text style features extracted from the news. Specifically, we construct a multi-view graph that encodes various relationships between news items from the same publisher, such as news topics and occasions in which they were released. Furthermore, we leverage a multi-layer Graph Convolutional Network in conjunction with jumping knowledge networks to model the multi-view graph and produce a publisher-oriented contextualized representation of news. Experimental results on two widely used fake news datasets, namely LIAR and Weibo21, demonstrate the effectiveness of our approach. Specifically, the PMGM model outperforms the state-of-the-art methods significantly. Overall, our proposed model unifies various heterogeneous features and information related to news based on a publisher-oriented approach, thereby offering a novel idea to enhance fake news detection.

Index Terms—Fake News Detection, Multi-View Graph, Representation Learning, Relation Graph,

I. INTRODUCTION

Currently, the proliferation of fake news poses a significant threat to the reliability and veracity of news. The public’s trust in the British government during the "Brexit" referendum and the fairness of the 2016 U.S. presidential election have been greatly undermined by the spread of fake news [1], [2]. As a result, there has been a growing interest in the NLP community in the development of fake news detection systems that can automatically assess the authenticity of a given news text [3], [4], [5], [6], [7].

Early research in this area focused on the manual engineering of features [3], [4]. Initially, researchers created comprehensive sets of hand-crafted features based on news content, user profiles, and news propagation paths. They then trained machine-learning classifiers to distinguish between true and false news. However, recent studies [6], [7], [8] have leveraged the success of deep learning and applied various neural network models, such as Convolutional Neural Networks (CNN) [9].

*Corresponding author
DOI reference number: 10.18293/SEKE2023-061

President Donald Trump:
Republican from New York
The PolitiFact scorecard: [63,114,51,37,61]
His over 800 news as follows:
On the VA: Over 300,000 veterans have died waiting for care.
We admit about 100,000 permanent immigrants.
Ted Cruz is mathematically out of winning the race.
...
I don't know anything about David Duke.

Speaker profile: President Donald Trump,
a Republican from New York, published a news about
health-care in a speech. His credit history is [63,113,51,37,61].
News text: On the VA: Over 300,000 veterans have died waiting for care.
[Short text]
Text style feature:
[#Words:12, #Nounphrases:2, …… #Certainty:0.09, #Subjectivity:0.83]

Fig. 1. Donald Trump’s profile and published news (blue rectangle) as well as the features used in this paper (red rectangle).
related work. Our model extracts 40 latent textual features as text style features given a piece of news text. It encodes the words in the news text and publisher profile using BERT, and then combines these features to represent the news text. Furthermore, we propose a multi-view news graph that is oriented towards the publisher. The nodes of the graph are news representations, and the edges model the relationships between news in different views, such as topics and occasions. The news representations are updated using Graph Convolutional Networks (GCN) and Jumping Knowledge Networks (JK-Nets) [16]. The representations from different views are fused using the attention mechanism [17]. Finally, a neural classifier is used to predict the labels of all news jointly. To demonstrate the effectiveness of our model, we conduct experiments on two fake news datasets, LIAR and Weibo21. The results show that PMGM significantly outperforms state-of-the-art methods.

II. RELATED WORK

A. Feature-Based Methods

Feature-based fake news detection methods rely on either hand-crafted linguistic or embedding features to extract information about the writing style or language used in news text [5], [13], [19], [8]. For instance, Castillo et al. [3] introduced a decision tree-based model that utilized an extensive range of features for identifying fake news on Twitter. Similarly, Yang et al. [4] extracted a broad set of features from microblog data to train a classifier for automatically detecting fake news on Sina Weibo. Rubin et al. [20] provided a conceptual overview of satire and humor and illustrated the unique features of satirical news to detect potentially misleading information. Wu et al. [21] used a propagation structure composed of 23 features in the hybrid support vector machines (SVM) method. Alternatively, Wu et al. [22] proposed a machine learning model that relied on time series fitting of tweet volume time characteristics. However, these methods heavily rely on manual feature engineering and may not capture enough features to achieve satisfactory performance when dealing with short news articles.

B. Relation-Based Methods

As the fields of deep learning and graph neural networks continue to advance, relation-based methods that exploit certain connections between news articles have come to dominate this domain [13], [14], [15]. Long et al. [5] utilized publisher profiles as a means of representing the attention factors between news articles to propose a hybrid LSTM model for detecting fake news. Karimi et al. [18] combined information from multiple sources to discriminate between different degrees of fake news by taking into account the relationships between them. Hu et al. [13] proposed a graph that incorporates publisher profiles for fake news detection using multi-depth graph convolutional networks (M-GCN). Mendoza et al. [23] analyzed the topology of retweeting networks and identified differences between rumor diffusion patterns on Twitter and traditional news platforms. Li et al. [24] combined objective information with subjective factors for rumor detection. Kwon et al. [25] introduced the Periodic External Shocks (PES) model, which combines a set of linguistic features with network structure to identify rumors. However, all of these methods utilize only homogenous relationships between news articles from a single perspective, without considering the possibility of heterogeneous relationships.

III. METHOD

A. Model Overview

We present a novel approach for detecting fake news, which we call the Publisher-Oriented Multi-View Graph Model (PMGM). As shown in Figure 2, our framework comprises three key components. Firstly, our model extracts 40 latent textual features as text style features from a given news article and its corresponding publisher profile. We encode both the news article and publisher profile using BERT, as detailed in Section III-B to obtain the enhanced contextualized representations of each news. These features are then combined to represent the news. Secondly, we construct a publisher-oriented multi-view graph, where nodes represent the news representations and edges are created based on the associations of topics and occasions between news articles from the same publisher, as discussed in Section III-C. Finally, we apply an attention mechanism to fuse the features of different views in the graph, and use a neural classifier to determine the labels of all news articles.

B. Enhance News Representation

We amalgamate the three characteristics of news text, publisher profile, and text style feature to form the publisher-oriented representation of a news item. Subsequently, we delineate the processing and embedding of the aforementioned features separately.

Encoding of News Text. To encode the news text, we utilize BERT as the embedding layer to extract the output vectors of all word vectors of all word tokens. The resulting global vector \( t_i \) corresponding to the [CLS] token represents the \( i \)th news.

Encoding of Publisher Profile. We use various publisher profile information, including party, publisher name, home state, credit history, social occasion and topic, to enhance the performance of the fake news classifier. For discrete publisher profile information, we directly use their encoded representation. For instance, the credit history [23,12,22,43,61] is encoded as \( s_1^d = [23, 12, 22, 43, 61] \). For text-format publisher profile information, we reorganize them using the template \( \text{job publisher name, a party from home state, published a news about topic in a social occasion} \) and then utilize BERT as the embedding layer to transform the information into feature vectors \( s_1^t \). Finally, we concatenate \( s_1^d \) and \( s_1^t \) to obtain \( s_i \), which represents the publisher profile of the \( i \)th news.

Extraction of Text Style Features. According to the definition of text style [26], a set of quantifiable characteristics (such as machine learning features) can effectively represent text content. Given a news text \( N \) to be verified, we represent it as a set \( d_k = \{ d_1^k, d_2^k, d_3^k, ..., d_k^k \} \) of \( k \) text style features, where \( d_k \) typically takes the form of a number. To adequately capture
and represent the style of news text, we synthesized various fake news detection papers to obtain 40 features \[26, 27\] as shown in Table I. We then include these features in our news representation. Finally, the representation of each news \( h_i \) is summarized as \( h_i = [t_i] \| s_i \| d_i \), where \( \| \) denotes concatenation.

### C. Publisher-oriented Multi-view Graph Modeling

**Multi-view Graph Design.** The news published by a single entity covers diverse topics and occasions, posing a challenge in explicitly modeling the correlation between all news solely based on a single viewpoint \[28\]. To address this, we propose a publisher-oriented multi-view news graph that leverages distinct publisher perspectives. We establish edges among news nodes of the same publisher based on shared topic or social occasion. We create three graphs, namely the publisher full-connected graph, topic-oriented graph, and occasion-oriented graph. The occasion devotes social occasions, such as TV shows, interviews, and election campaigns, where news is published. For the publisher full-connected graph, we define the sparse adjacency matrix \( A^1 \) as follows:

\[
A^1_{ij} = \begin{cases} 
1 & \text{if } i, j \text{ from same publisher} \\
0 & \text{otherwise} \end{cases} \tag{1}
\]

For the topic-oriented graph, we define the sparse adjacency matrix \( A^2 \) as follows:

\[
A^2_{ij} = \begin{cases} 
0.5 & \text{if } i, j \text{ have the same topic from same publisher} \\
0 & \text{otherwise} \end{cases} \tag{2}
\]

For the occasion-oriented graph, we define the sparse adjacency matrix \( A^3 \) as follows:

\[
A^3_{ij} = \begin{cases} 
0.2 & \text{if } i, j \text{ have the same occasion from same publisher} \\
0 & \text{otherwise} \end{cases} \tag{3}
\]

where \( i \) and \( j \) are the indices of different news.

**Graph Embedding.** Graph Convolutional Networks (GCNs) have shown significant promise in achieving generalization in various tasks. Our work builds upon the GCN module. At the \( i \)th layer, the GCN module takes the graph adjacency matrix \( A_i \) and hidden representation matrix \( H_i = \{h_1, h_2, ..., h_{|D|}\} \) with \( |D| \) news articles as input. The GCN module then outputs a hidden representation matrix \( H_{i+1} \in \mathbb{R}^{n_i \times d_{i+1}} \), which can be described as:

\[
H_{i+1} = \text{ReLU} \left( \tilde{D}_i^{-\frac{1}{2}} \tilde{A}_i \tilde{D}_i^{-\frac{1}{2}} H_i \theta_i \right) \tag{4}
\]

where adjacency matrix with self-loop \( \tilde{A}_i = A_i + I \), \( \tilde{D}_i \) is the degree matrix of \( \tilde{A}_i \), and \( \theta_i \in \mathbb{R}^{d_i \times d_{i+1}} \) is a trainable weight matrix.

To enhance the structure-aware representation of a node, we utilize a combination of GCN and jumping knowledge networks (JK-Nets) based on the approach proposed by Xu et al. \[16\]. JK-Nets involve the direct transfer of each layer’s
representation to the final layer of the network, rather than solely passing it to the next layer of the convolutional network. This facilitates the aggregation of information from distinct receptive fields at the last layer, and allows the training process to maximize features by determining the receptive field size for each node. We obtain the output $h^i_l$ for each graph node as follows:

$$h^i_l = \max(h_1, ..., h_i)$$  \hspace{1cm} (5)

where $h_i$ is the node representation of the $i$th GCN layer for the node $h$ of $H_{lc}$.

**Learn Publisher-oriented Representation.** After encoding both the node features and the graph structure of a multi-view graph in an end-to-end manner, we aggregate the multi-view information using an attention mechanism to form an updated representation. Specifically, we assign an attention score $u_i$ (where $1 \leq i \leq m$) to the node representation of each view, which is then normalized using the softmax function. The publisher-oriented representation $h^u_i$ for each news item is obtained by computing the weighted summation of the individual view representations $h^i_v$, where the weights are determined by the attention scores.

$$u_i = \tanh (W_i h^i_v + b_i); \alpha_i = \frac{\exp(u_i)}{\sum_{i=1}^{m} \exp(u_i)}; h^i_v = \sum_{i=1}^{m} \alpha_i h^i_v$$  \hspace{1cm} (6)

**D. Fake News Prediction**

Research by Dou et al. [29] has shown that incorporating updated news representation with the original news can enhance the performance of fake news detection. Prior to inputting the final representation into the classifier, we concatenate the [CLS] representation of the original news with the publisher-oriented news representation. Subsequently, we apply a softmax classifier to predict the truthfulness label of the news.

$$\hat{y}_i = \text{softmax} \left(\text{ReLU} \left( [h^u_i | \alpha_i] W_{tp} + b_{tp} \right) \right)$$  \hspace{1cm} (7)

Where $W_{tp}$ and $b_{tp}$ represent the parameters of the output layer, $h^u_i$ and $\alpha_i$ represent the updated news representation and the [CLS] representation of the original news, respectively. Our model is trained using the cross-entropy loss during the training phase, which can be formalized as:

$$\mathcal{L} = - \sum_{i \in D} y_i \ln \hat{y}_i$$  \hspace{1cm} (8)

Where $y_i$ represents the ground-truth label for the truthfulness of the $i$th news article, and $\hat{y}_i$ represents the predicted distribution for the truthfulness label of the same article. Our objective is to minimize the loss function $\mathcal{L}$ for the purpose of detecting fake news.

**IV. EXPERIMENTAL SETTINGS**

**A. Dataset**

We evaluated our model using two datasets, LIAR [6] and Weibo21 [30], which contain instances of fake news. The LIAR dataset includes 12,800 human-labeled short news items with six fine-grained labels that indicate the degree of truthfulness, including pants-fire, false, barely-true, half-true, mostly-true, and true. The dataset’s label distribution is as follows: 1,050 for pants-fire, and a range of 2,063 to 2,638 for the other labels. On the other hand, the Weibo21 dataset is a Chinese multi-domain dataset with 4,488 fake news items and 4,640 real news items. These datasets are unique in that they include several metadata features, such as the topic, publisher, job, state, party, and total credit history count of the news publisher.

As standard practice in machine learning, we split the datasets into training (80%), validation (10%), and testing (10%) sets. The LIAR dataset has 3,308 publishers, 144 news topics, and 302 occasions. To provide an overview of the publishers, topics, and occasions in this dataset, we present the top-5 most frequent publishers, topics, and occasions in Table II.

**B. Implementation Details**

During the text processing stage, the initial step involves cleansing the text information by eliminating redundant expressions and symbols, standardizing the case, and so forth. In this study, we utilized BERT-base to acquire 768-dimensional embeddings for each news. The hidden unit dimensions in GCN were established as [768, 768], while the number of GCN layers was 6. A dropout rate of 0.5 was specified. To optimize all parameters, we employed the Adam optimizer, coupled with a weight decay strategy, to train the model for 80 epochs. To ensure a level playing field for comparisons, we adopted the same evaluation metrics that were utilized in previous research, namely Accuracy and F1-measure (F1).

**C. Baselines**

We compared our proposed models with existing fake news detection models, including state-of-the-art models, on both the LIAR dataset and Weibo21 dataset. The models that we compared with are as follows: 1) CNN-WangP [6]: A hybrid CNN that integrates both text and contextual information to detect fake news. 2) MMFD [18]: A multi-source, multi-class fake news detection model that employs multiple sources of information to detect fake news across various classes. 3) LSTM-Attention [5]: A hybrid LSTM that accounts for word importance using an attention mechanism. 4) FT+BERT [19]: A fine-tuning technique based on the BERT pre-trained language model, which we utilized in our study. 5) FakeBERT [8]: A model that combines various parallel blocks of a single-layer deep CNN, each with different kernel sizes and filters, with the BERT. 6) M-GCN [13]: A semi-supervised fake

**TABLE II**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Barack Obama</td>
<td>611</td>
<td>healthcare</td>
<td>474</td>
<td>news release</td>
<td>309</td>
</tr>
<tr>
<td>Donald Trump</td>
<td>343</td>
<td>taxes</td>
<td>356</td>
<td>interview</td>
<td>286</td>
</tr>
<tr>
<td>Hillary Clinton</td>
<td>297</td>
<td>education</td>
<td>309</td>
<td>press release</td>
<td>282</td>
</tr>
<tr>
<td>Mitt Romney</td>
<td>212</td>
<td>elections</td>
<td>304</td>
<td>speech</td>
<td>259</td>
</tr>
<tr>
<td>John Mccain</td>
<td>189</td>
<td>immigration</td>
<td>303</td>
<td>TV ad</td>
<td>222</td>
</tr>
</tbody>
</table>
news detection method that leverages text content as node features and publisher profiles to build a graph. 7) MTFake [14]: A multitask learning model that categorizes news articles collected from the web as either fake or not. 8) MMFake [15]: A multitopic and multitask fake news detection model that addresses the limitations of fusing different topics. 9) PMGM: Our proposed publisher-oriented multi-view graph modeling (PMGM) framework.

V. RESULTS AND ANALYSIS

A. Comparisons with Baselines

Table III presents the primary experimental results, along with the inter-model differences. Notably, the proposed PMGM model demonstrates significantly better performance than the state-of-the-art model on both the LIAR dataset and Weibo21 dataset. It is clear from the table that the relation-based methods (M-GCN, MTFake, MMFake, and PMGM) significantly outperform the feature-based methods (CNN-WangP, MMFD, LSTM-Attention, FT+BERT, and FakeBERT). This is largely attributed to the relation-based methods’ ability to learn semantically rich representations of news, which enable them to capture more effective features. This underscores the importance of studying relation-based methods for fake news detection.

Specifically, our proposed PMGM method outperforms the MTFake and MMFake methods for two main reasons. Firstly, these methods do not delve into the representations of news texts, which are crucial in short news texts. Secondly, while MMFake only considers a relationship between news, PMGM considers both publisher-oriented multiple relations between news. This highlights the effectiveness of enhancing the representation of short news texts and employing a publisher-oriented multi-view graph model for fake news detection.

B. Ablation Studies

We conducted a series of ablation studies on key parts of the MVAN in order to determine the relative importance of each module, using the LIAR dataset. The experimental results of this comparison are presented in Table IV. Our findings indicate that when the PMGM model removes the publisher profiles, performance drops by approximately 25%. This can be attributed to the fact that publisher profiles contain credit history, which is a statistical dataset collected from previous statements of publishers and not readily available. Furthermore, when compared to MMFake without credit history, our model shows an improvement in accuracy of about 1.67%. We also discovered that removing each of the publisher-oriented multi-view components led to a drop in performance of approximately 2.4%, 1.7% and 0.6% respectively, demonstrating that multi-view mechanisms have a significant impact on model performance. Additionally, we observed that not using the [CLS] of original news led to a 1% drop in model performance. This indicates that combining the updated news representation with the original news can significantly enhance the performance of fake news detection.
C. Effect Of The Multi-View Graph Modeling

To explore the impact of a multi-view module on fake news detection, we demonstrate the significance of the module in Figure 4. The results indicate that the model achieves higher accuracy in classifying false, barely-true, half-true, and mostly-true news when the multi-view module is utilized, and the model’s performance in correctly classifying labels declines when the module is removed. Hence, our approach, PMGM, leverages a publisher-oriented multi-view graph to aid the detection task, as the limited information present in short news content makes it difficult to obtain sufficient representations during model learning, leading to poor model performance.

VI. Conclusion

In this paper, we propose a novel deep learning model, PMGM, for detecting fake news. The model combines a rich news representation with a multi-view of the publisher to capture important hidden clues and information in both the news text and its publisher. Our evaluation using two public datasets demonstrates that PMGM outperforms existing methods in fake news detection. Furthermore, we anticipate that utilizing the publisher-oriented multi-view graph model will also prove advantageous in other text classification tasks, such as sentiment and topic classification. In future work, we intend to expand our research in the following areas: (1) constructing a larger and more current dataset that includes additional publisher and user profiles and propagation data, (2) incorporating publisher and user information using more sophisticated methods, and (3) integrating additional publisher information into PMGM to achieve more robust evaluations of the fake news model.

Acknowledgements

This work is supported by the National Key Research and Development Program of China No. 2017YFC1200500, the Research Foundation of Ministry of Education of China No. 181ZD015.

References

PSRL: A New Method for Real-Time Task Placement and Scheduling Using Reinforcement Learning

Bakhta Haouari ✪ Rania Mzid ✫ Olfa Mosbahi ✩

* ISI, University Tunis-El Manar, 2 Rue Abourraihan Al Bayrouni, Ariana, Tunisia
✩ LISI Lab INSAT, University of Carthage, INSAT Centre Urbain Nord BP 676, Tunis, Tunisia
✚ CES Lab ENIS, University of Sfax, B.P:w.3, Sfax, Tunisia

Abstract

Modern real-time system development methodologies describe a stage in which application tasks are deployed onto an execution platform. The deployment process is divided into two steps: (i) task placement on processors and (ii) task scheduling to determine their execution order. The overall performance of the deployment model depends on the two steps, which are interdependent. In this paper, a new method based on reinforcement learning techniques, called PSRL, is proposed. PSRL explores all the feasible placements in the first step. In the second step, an optimal schedule is considered for each feasible placement. PSRL generates the optimal deployment, which corresponds to the placement and scheduling that minimize task response times. Application to case studies shows the applicability and quality of the obtained solutions when compared to related work.

1 Introduction

Today, embedded systems can be found in a wide range of applications, including avionics, trains, and medical equipment. Embedded systems are frequently real-time as they must perform certain tasks in a specific amount of time (i.e., deadline); failure to satisfy the timing constraints might be essential for human safety [1]. Currently, real-time embedded systems are becoming more complex, requiring more computational power. As a result, to be executed, a system’s functionalities (i.e., tasks) may be distributed on different execution units called processors.

The development of Real-Time Embedded Systems (RTES) requires the definition of the deployment model. The deployment stage involves two main steps: placement and scheduling [2]. The task placement problem is concerned with task-to-processor mappings. The scheduling problem, on the other hand, aims to define the execution order of tasks in each processor. To guarantee the respect of real-time properties, design-time verification using schedulability analysis techniques [3] is typically employed.

Producing a deployment model has been proved to be NP-hard problem even for small-size systems [3]. Due to their inherent complexity, placement and scheduling problems should be automated by solving an optimization problem with respect to real-time constraints. This problem has been largely considered in the literature. Some works consider partially the problem by concentrating solely on the placement or scheduling issue. In [4], the authors use genetic algorithms (GA) to solve the task placement problem in distributed systems to minimize communication cost. The authors in [5] provide a Mixed Integer Linear Programming (MILP)-based technique to minimize the blocking time between tasks in the placement phase. Producing the optimal scheduling of tasks is not considered in this work; however, the authors use the Rate-Monotonic algorithm [6] for priority assignment and focus on minimizing the number of tasks to improve the overall performance of the system. The scheduling problem is addressed in [7] where a new algorithm based on Reinforcement Learning (RL) is proposed for global fixed-priority task assignment in multi-processor real-time systems. In [8], the authors propose a new scheduling algorithm that maximizes the magnitude of safety margins while respecting the engineering constraints. In [9], new scheduling policies are proposed for heterogeneous platforms equipped with a hardware accelerator. When dealing with both problems (i.e., placement and scheduling), studies in the literature consider, in general, sub-optimal staged approaches. Existing approaches use optimization techniques to optimize independently the placement and the scheduling steps. For instance, in [2], the authors propose a GA-based approach where they minimize the number of processors in the placement step and the response time of tasks in the scheduling step. However, the authors claim that the obtained solution is near-optimal since the optimization performed in the first step can hide a better solution in the second phase.

To address this problem, we propose in this paper an RL-based technique called PSRL (i.e., Placement and Scheduling using Reinforcement Learning). The PSRL method defines two steps: The first stage involves an exhaustive RL-based search of all feasible placements and which serves as input to the second stage. The second step determines an RL optimal scheduling for each placement. The global optimal scheduling is then the best among all the alternatives. The optimal scheduling in this paper is the one that optimizes the response time of tasks. The placement and scheduling

DOI reference number: 10.18293/SEKE23-178
problems are thus modeled as a Markov Decision Problem (MDP) [6] and the Q-learning algorithm [10] is used in the placement and scheduling steps. The originality of this paper is the use of Q-learning algorithm, typically used to generate an optimal solution, in the placement phase to conduct an exhaustive search of feasible solutions through the exploitation of the Q-table. As a result, the scheduling step explores all the optimal solutions without being constrained by a given placement, thus avoiding producing sub-optimal solutions.

The rest of the paper is organized as follows: Section 2 gives the system formalization. Section 3 describes the proposed PSRL method and details RL algorithms. Section 4 illustrates experimentation, and Section 5 concludes the paper and discusses future directions.

2 System formalization

The placement problem considers as inputs: (i) the task model that we denote by $\tau$ in this work. We assume that this model is composed of $n$ synchronous, periodic, and independent tasks (i.e., $\tau = \{T_1, T_2, \ldots, T_n\}$). Each task $T_i$ is characterized by static parameters $T_i = (C_i, p_{ri}, d_i)$ where $C_i$ is an estimation of its worst case execution time, $p_{ri}$ is the activation period of the task $T_i$, and $d_i$ is the deadline that represents the time limit in which the task $T_i$ must complete its execution, (ii) and the hardware model that we denote by $P$, represents the execution platform of the system. We assume that this model is composed of $m$ homogeneous processors (i.e., $P = \{P_1, P_2, \ldots, P_m\}$).

The placement step produces a set of possible placements that we denote $PM$. Each placement $PM_k \in PM$ defines a way to place tasks among the different processors in the hardware model. For RTES, the placement model $PM_k$ must be feasible. Feasibility means that the placement of the real-time tasks on the different processors must guarantee respect for the timing requirements of the system. In that context, Audsley [3] developed a necessary and sufficient schedulability test. This test is based on the computation of the processor demand factor $U_p$ and is defined as follows:

$$U_p = \sum_{i=1}^{n} \frac{C_i}{d_i} \leq n(2^\frac{1}{n} - 1)$$  \hspace{1cm} (1)

For each feasible placement, $PM_k$, the scheduling step defines, in each processor, the execution order of tasks to which they are assigned. As a result, a deployment model $DM_k$ is produced. We denote by $DM$ the set of deployment models produced in the scheduling step. Each task $T_i$ in the deployment model $DM_k \in DM$ will be mapped to a particular processor and will be assigned a priority $p_i$. Priority assignment in the scheduling stage must guarantee the feasibility checked in the placement step. Real-time feasibility at this level guarantees that the response times of the different tasks are lower or equal to their deadlines (i.e., $\forall T_i \in \tau, RT_i \leq d_i$). The response time of a task $T_i$ is given in [3], and is computed according to the following expression:

$$RT_i = C_i + \sum_{s \in hp(i)} \left( \frac{RT_s}{p'r_s} \right) C_s$$  \hspace{1cm} (2)

Where $hp(i)$ refers to the tasks with priorities higher or equal to the priority of the task $T_i$.

An optimal deployment model $DM_{*k}$ is one that minimizes the $Sum\ RT\ ratio$ defined by [9] as a measure of performance for scheduling models, which is computed as follows:

$$\sum_{i=1}^{n} \frac{RT_i}{d_i}$$  \hspace{1cm} (3)

We identify by $DM^*$ the deployment model that has the lowest $Sum\ RT\ ratio$ among all produced $DM_{*k}$, such as

$$\sum_{T_i \in DM^*} \frac{RT_i}{d_i} = \min_{DM_{*k} \in DM} \sum_{T_i \in DM_{*k}} \frac{RT_i}{d_i}$$  \hspace{1cm} (4)

3 PSRL description

In this section, we describe the proposed method for real-time task placement and scheduling. The PSRL method is schematically described in Figure 1; the specific steps are shown in Algorithm 1. The PSRL technique considers as entries: (i) the task model, which describes the application functions, and (ii) the hardware model, which describes the execution processors. The initial stage is to generate possible feasible placements to the given problem. After producing all solutions, the second phase iterates on each placement to define the optimal scheduling. In fact, the second phase generates one deployment model $DM_{*k}$ for each placement $PM_k$ which minimizes the $Sum\ RT\ ratio$ (see expression 3). Once all of the deployment models for each placement have been generated, the PSRL algorithm selects the optimal solution from among those generated ($DM^*$) (as defined in the expression 4).

3.1 Step 1: Generate feasible placement models

As previously mentioned, the objective of this step is to produce placement models for a given task and hardware models. As shown in Figure 1, when no solution is found, the designer has to adjust the parameters of the entry models. Otherwise, this step generates all feasible placement models. In this step, we are particularly referred to the Q_learning algorithm (i.e., a free model RL algorithm) for the exploration of the design space. Indeed, during its processing, Q_learning generates a Q_table as a workspace where it stores all its knowledge about the task placements. At the end of the agent learning, the Q_table is used to generate a unique solution, the optimal one. In this paper, since
Algorithm 1: PSRL Algorithm

**Input:** \( \tau \): List of tasks
\( P \): List of processors

**Output:** \( DM^* \): The optimal deployment model

**Notations:**
\( PM \): Feasible placement models
\( DM \): Optimal deployment models
\( DM^* \): Best deployment model

```
PM ← Generate feasible placement models (\( \tau, P \));
foreach \( PM_k \in PM \) do
    \( DM^*_k \) ← Generate Optimal Scheduling (\( PM_k \));
    \( DM \) ← Update(\( DM^*_k \));
end
DM^* ← Select Best Solution (\( DM \));
return DM^*;
```

optimal placement does not guarantee an optimal deployment model, we propose to use Q-learning for an exhaustive search of the feasible placements (the optimal and the sub-optimal ones). In fact, we believe that sub-optimal placements can hide good scheduling, even the optimal one. As a result, we use the Q-table to extract all the placement models. The final states in the Q-table refer to possible placement models that may be feasible or not following the initial designer-specified constraints. Application of RL techniques to the placement problem requires the refinement of its key elements as the following:

**State:** It is a snapshot of the system state at time step \( t \), it is represented by the collection of tasks already placed on the processors with respect to their deadlines, as well as the list \( L_t \) of tasks that have not yet been placed

**Decision epoch:** Matches the placement of all the tasks on the different processors

**Agent:** It is the decision maker, it has to be learned and then used to decide a processor for a given task

**Reward (R):** It is the award given by the system to the agent in return to the agent’s action and it is computed as Equation 5.

\[
R = \begin{cases} 
U_j - U_i & \text{When there is enough space on } P_j \text{ to support } T_i, \\
-n & \text{Otherwise}
\end{cases}
\]  

Where \( U_j \) denotes the utility of processor \( P_j \) and \( U_i = C_i / d_i \) denotes the utility of the task \( T_i \). Algorithm 2 describes the first step. As an initial step, the algorithm builds the Q-table then the agent starts its learning through the assignment of a given task to a processor. If the assignment helps to yield a feasible placement, the agent is rewarded; if not, it is penalized. Equation 5 summarizes the reward compute. The process of placement selection is iterated until the Q-table values become invariable. At this step, the totality of the system’s states are visited, and the majority of solutions are built in the Q-table final states, i.e states where all the tasks are placed. Note that a feasible placement is one in which each task from the task model is assigned to a processor with respect to the processor utility constraint (see equation1).

3.2 Step 2: Generate Optimal Scheduling

The set of possible placement models generated in the first stage is used as input for the second. Q-learning is then reused in the second stage to build an optimal deployment model \( DM_k \) for each placement \( PM_k \). For the scheduling problem, key RL elements are also refined as the following:

**State:** At a time step \( t \) a system state \( S_t \) is represented by the set of tasks already ordered and the ones waiting for a priority assignment

**Decision epoch:** Matches the ordering of all the tasks on a given processor

**Agent:** The decision maker, it chooses the action to move from \( S_t \) to \( S_{t+1} \) will maximizing its reward following a policy \( \pi \)

**Reward (R):** The bonus that the agent attempts to maximize or the penalty that the agent attempts to avoid following the execution of an action. In the scheduling problem, the agent must select the job that generates the minimum Sum RT-ratio (see equation3). However, because the agent seeks to maximize R, R is defined as the inverse of the RT-ratio and is calculated as:

\[
R = \begin{cases} 
1 / RT_i & \text{when } RT_i \leq d_i, \\
-m & \text{Otherwise}
\end{cases}
\]  

Figure 1: PSRL overview
The RL model for the scheduling stage is described in Algorithm 3. After creating the scheduling Q-table, the agent is trained how to select the task to be ordered with the aim to maximize its profit as described in the expression 6. This phase is repeated for each processor in the current PMk to finally produce the optimal deployment model DM∗k.

The proposed algorithms (Algorithm 2 and Algorithm 3) have some initialization parameters, such as α, which represents the learning rate to moderate the speed of learning and the update of Q-values we assume α = 0.5, and γ, which represents the discount factor that quantifies the importance given to future rewards (in our approach we consider that future task placement are important thus we attribute an enough great value to γ = 0.9). To choose an action (i.e., for the placement or scheduling), Q-learning uses an ε-greedy policy. ε-greedy policy is an efficient random approach that selects with a probability ε a random action and with a probability (1-ε) the action with the highest estimated reward Q(S, a). An ε of zero means that the agent will never explore new states (i.e., choose action with the best Q-value), whereas an ε of one forces the agent to only explore (i.e., choose only random action). As a result, a well-studied value of ε is necessary to capture a trade-off between exploration and exploitation.

4 Evaluation

This paper considers two case studies to assess the applicability and effectiveness of the proposed PSRL method. The first is a subsystem of Continental AG’s Cruise Control System (CCS), which is used on AUTOSAR-compliant architectures [12] and allows a car’s speed to be maintained regardless of the surface form (flat or sloping) on which it is driving. The second case study is the Unmanned Air Vehicle (UAV), which is an autonomous plane with a camera installed in a vehicle (UA V), which is an autonomous plane with a camera.

The results discussed in Section 4 confirm the effectiveness of the proposed approach, which is shown to produce good performance with minimal computational overhead. The proposed algorithms are able to adapt to new environments and learn from experience, thereby improving their performance over time.
Table 1: UAV Taks model description

<table>
<thead>
<tr>
<th>Task</th>
<th>( C_i ) (m)</th>
<th>( pr_i ) (m)</th>
<th>( di_i ) (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 )</td>
<td>1</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>1</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>( T_3 )</td>
<td>5</td>
<td>125</td>
<td>30</td>
</tr>
<tr>
<td>( T_4 )</td>
<td>5</td>
<td>125</td>
<td>80</td>
</tr>
<tr>
<td>( T_5 )</td>
<td>1</td>
<td>50</td>
<td>7</td>
</tr>
</tbody>
</table>

five periodic tasks for UAV with their real-time parameters.

Table 2: CCS Taks model description

<table>
<thead>
<tr>
<th>Task</th>
<th>( C_i ) (m)</th>
<th>( pr_i ) (m)</th>
<th>( di_i ) (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 )</td>
<td>3</td>
<td>125</td>
<td>20</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>3</td>
<td>125</td>
<td>80</td>
</tr>
<tr>
<td>( T_3 )</td>
<td>2</td>
<td>125</td>
<td>20</td>
</tr>
<tr>
<td>( T_4 )</td>
<td>4</td>
<td>125</td>
<td>80</td>
</tr>
<tr>
<td>( T_5 )</td>
<td>2</td>
<td>125</td>
<td>20</td>
</tr>
<tr>
<td>( T_6 )</td>
<td>4</td>
<td>125</td>
<td>80</td>
</tr>
<tr>
<td>( T_7 )</td>
<td>3</td>
<td>125</td>
<td>80</td>
</tr>
<tr>
<td>( T_8 )</td>
<td>2</td>
<td>125</td>
<td>80</td>
</tr>
</tbody>
</table>

Since the objective of the first step is to perform an exhaustive search of feasible placements, we perform, as a first evaluation, a sensibility analysis of the number of feasible placement models. We maintain a set of experiments on the epsilon value (\( \epsilon \) in Algorithm 2), which guides the exploration/exploitation process to fill the \( Q \) table and subsequently identify the feasible placements. While the value of epsilon is increased at each step, the placement generation algorithm (Algorithm 2) is run and the number of feasible placement models is tracked. We remark that the number of placement models increases from one step to another until it converges and reaches a maximum value (i.e., the number of feasible placement models for the considered application). However, this comes at the expense of the number of iterations required for the agent to reach convergence. In fact, a low epsilon value forces the agent to rely on its limited experience, decreasing the possibility of many states in the \( Q \) table being visited and, as a result, reducing the number of feasible solutions generated. A large epsilon value allows the agent to visit more states in the \( Q \) table, but also causes the time of convergence to be delayed. A trade-off between the episode number and the epsilon value is required to generate the set of possible placement models in a reasonable number of iterations.

Figure 3 and Figure 2 show the evaluation results for respectively the UAV and the CCS case study. Figure 3 shows two curves. The green corresponds to the feasible placement models number for various epsilon values, whereas the blue corresponds to the total number of placement possibilities (i.e., feasible and non feasible placement models). As the number of placement models is the same for epsilon \( \epsilon \in [0.6, 1] \), it is unnecessary to run PSRL with epsilon greater than 0.6 for the UAV case study. An epsilon value greater than 0.6 results in more useless episodes. We also note that the number of feasible placement models is close to 1200, which is relatively low when compared to the total number of placement models. Indeed, this is explained by the fact that, due to the real-time parameters considered for the UAV case study, solutions that result in tasks running in one processor (\( P_1 \) or \( P_2 \)) are all rejected. Figure 2, in contrast to Figure 3, shows only one curve. Indeed, for the CCS case study, the real-time parameters considered lead to a scenario in which all possible placement models are feasible. This explains why we get the same curves.

Figure 3: UAV placement models

We conduct a second set of experiments on the two case studies, where we set the epsilon value to 0.6 for the UAV and to 0.5 for the CCS, respectively (i.e., results of the first experiments). The goal of this evaluation is to compare the quality of the derived optimal model to the approach described in [2], where the deployment model is generated in two steps: the first seeks to minimize the number of processors, while the second aims to decrease task response time. Figures 4 and 5 demonstrate, for the CCS and UAV case studies, the set of deployment models represented by the \( \text{Sum RT-ratio} \) values for all viable placement models, with extreme values matching the best schedule if it is a global minimum and the worst one if it is a global maximum. The figures show that the optimal solution for both case studies is far from matching the processor number minimization [2], thus a thorough search of all conceivable
placement is quite important for absolute optimal schedule generation. For the UAV case study, the optimal deployment model produced by the PSRL method is \( DM^* = \{P_1\{T_3, T_1, T_5, T_2\}, P_2\{T_4\}\} \), where \( T_1, T_2, T_3, \) and \( T_5 \) are assigned to the processor \( P_1 \) and \( T_3 \) has the highest priority value, however \( T_2 \) has the lowest one. \( T_4 \) in turns is assigned to the processor \( P_2 \). For the CCS case study, the optimal deployment model produced by the PSRL method is \( DM^* = \{P_1\{T_4, T_2, T_7, T_6\}, P_2\{T_3, T_5, T_8\}\} \), where \( T_1, T_2, T_6, \) and \( T_7 \) are assigned to the processor \( P_1 \) and \( T_1 \) has the highest priority value, however \( T_6 \) has the lowest one. Tasks \( T_3, T_4, T_5, \) and \( T_8 \) are placed in the processor \( P_2 \) where \( T_4 \) is assigned the highest priority and \( T_8 \) the lowest one.

5 Conclusion

In this paper, we propose a new method called PSRL for real-time task placement and scheduling. The proposed solution was based on reinforcement learning techniques, particularly the Q-learning algorithm, to solve the deployment problem for real-time systems in two stages. In the first stage, placement, an exhaustive search for all feasible solutions was performed. For each feasible placement, the scheduling stage produces the deployment model, which minimizes the response times of tasks. The best solution among all the ones generated is considered an optimal deployment for the given problem. PSRL was tested on two case studies with different properties, and the results prove the efficiency of PSRL compared to related work. In future work, we aim to extend the proposed approach by considering more complex systems. In addition, we intend to consider other optimization techniques. Thus, an extension of this technique to be multi-objective will be addressed.

References

A Triplet Network Approach for Chinese Confusing Text Classification

Rui Xu¹, Cheng Zeng¹,², Yu Jin Liu¹, Peng He¹,², Min Chen¹
¹School of Computer Science and Information Engineering, Hubei University, Wuhan, China
²School of Artificial Intelligence, Hubei University, Wuhan, China
³School of Cyber Science and Technology, Hubei University, Wuhan, China

Abstract—The pre-trained model in the Chinese text classification task has made significant progress. However, there is a lot of semantically ambiguous and confusing text in the Chinese text, which has a negative impact on the classification model. A triplet network approach for Chinese confusing text classification is proposed to address this problem. This method improves the traditional triplet network's way of randomly constructing sample combinations, compares the feature similarity between the screened confusing text, straightforward text, and ordinary text, and improves the clustering effect of Chinese text features. At the same time, embedding text and label are jointly learned in the same latent space to learn the similarities and differences between texts of the same category and texts of confused categories. Experiments on multiple Chinese text classification datasets demonstrate the negative impact of confusing text on model accuracy and verify the method's effectiveness in this paper.

Keywords—text classification; triplet network; confusing text

I. INTRODUCTION

As a semantic language, Chinese differs from a morphological language (e.g., English). Changing a small amount of characters will lead to a massive difference in Chinese semantics. This special nature makes the Chinese language highly complex and diverse, often resulting in confusing text that is difficult to learn. The personality characteristics of Chinese confusing text are not evident, usually contain more confusing words, and the feature similarity with the confused category is high, so it is difficult to distinguish accurately. As shown in Fig. 1, the most confusing text is concentrated around the classification decision line, and the model cannot effectively identify the ground-truth class.

The confusing text belongs to the research category of the hard sample, and a series of research on the hard sample is currently carried out in the field of computer vision[1-2]. Wang et al. [3] used data augmentation technology to balance the data volume of the hard sample and easy sample, which improved the model performance to a certain extent. Shrivastava et al. [4] proposed a hard example mining algorithm to add the screened hard sample to a new training batch for training, but this method is generally used for mining hard negative examples. Compared with the processing of the dataset, Jiang et al. [5] changed the model structure. They used the supervised contrastive learning mode to optimize the loss of hard negative examples and increase the distance between different classes. In fact, the essence of processing confusing text in the hard sample is to reduce the distance of the same category and expand the distance of the confusing category. In pedestrian re-identification, Cai et al. [6] used the characteristics of the triplet network to effectively draw similar samples closer and push away dissimilar samples.

However, using the traditional triplet network to construct sample combinations randomly, there are still noticeable differences between the feature vectors of the same class in Chinese texts. So, we propose based on a triplet network for confusing text in Chinese text classification. This method suppresses confusing text from deviating from the gold label by learning text feature information and label features in the same latent space. Moreover, improve the way of randomly constructing sample combinations in the traditional triplet network, let the confusing text select the positive and negative texts from the straightforward texts to form a triplet sample combination. On this basis, add a negative example selected from ordinary text to further improve the model's generalization ability and pass the obtained sample combination to the triplet network for targeted training. Let the model dig deep into the feature coding differences between the confusing text and the straightforward text in Chinese predictions to improve the classification effect of the model.

II. RELATED WORK

A. Chinese Confusing Text

Chinese confusing text restricts the performance of existing models in text classification tasks to a certain extent, and the confusing text is shown in Table 1. In Chinese text classification, the confusing text tends to be confused due to vague opinions or more confusing words. A thorny issue is how to effectively avoid the negative impact of confusing text on the model.
TABLE I. CONFUSING TEXT EXAMPLE

<table>
<thead>
<tr>
<th>Confusing text</th>
<th>True label</th>
<th>Wrong label</th>
</tr>
</thead>
<tbody>
<tr>
<td>科研人员发现癌症预警和诊断方法</td>
<td>Health</td>
<td>Technology</td>
</tr>
<tr>
<td>呵！这一次成功真的不容易</td>
<td>Happiness</td>
<td>Surprise</td>
</tr>
<tr>
<td>有点慢，其他的还好</td>
<td>Positive</td>
<td>Negative</td>
</tr>
</tbody>
</table>

Prabhakar et al. [7] combined the attention mechanism and the Focal Loss function, and some text category confusion has been improved to a certain extent. Xu et al. [8] used a graph neural network combined with an attention mechanism to learn the feature differences between confusing legal texts. It has a good effect on legal texts but ignores the relevance between legal entries of the same type. Therefore, how to further cluster texts of the same class while expanding the distance of confusing class is an essential idea for studying confusing text. Surprisingly, in pedestrian re-identification, the triplet network has outstanding performance in dealing with pedestrians’ overlapping and confusing problems, which can solve this problem very well. Therefore, we use the triplet network to train confusing text and optimizes and improves it according to the characteristics of Chinese texts.

B. Triplet Network

The triplet network is developed from the Siamese network. Chopra et al. [9] proposed the Siamese network to solve the problem in the field of face recognition that the model cannot accurately identify personality characteristics due to similar face structures. The Siamese network uses two neural networks with the same structure and shared weight and inputs a positive face picture and a negative face picture into the model. After calculation, the feature similarity of the two samples can be obtained, which can be effectively increased through training—the distance between different classes.

However, the Siamese network is more sensitive to anchor samples, and the error in distinguishing different individuals in the same group category is relatively large. In this regard, Hoffer proposed that the triplet network [10] uses anchor samples, positive and negative examples to form a training group and uses the neural network model shared by three weight to extract input features for triplet loss calculation, which effectively solves the problem in the Siamese network. For problems with poor individual recognition ability in the same class, the triplet network performs better than the Siamese network on multiple tasks. Especially for pedestrian re-identification [11], the triplet network can track the trajectories of overlapping pedestrians very well. The triplet network can also be used to compare the feature information of confusing text with other texts to mine rich semantic information. Additionally, when Chen et al. [12] added a negative sample to the traditional triplet training criteria when constructing triplet combinations, they found that it could better reduce intra-class differences and increase inter-class differences. Inspired by this work, we improve the traditional method of randomly selecting samples by triplets. We select positive examples, negative examples, and confusing texts from straightforward texts to form a triplet sample group. At the same time, adding a negative example selected from ordinary texts can improve the model's generalization ability. Therefore, while measuring different categories of texts, Our method focuses on the samples selected from straightforward texts to dig out the standard text features of easy and hard samples.

III. APPROACH

Fig. 2 shows our triplet network's main idea for Chinese confusing text. The main steps include 1) fused label embedding in standard classification, 2) screening confusing text and straightforward text, and 3) training confusing text based on the triplet network.

A. Standard Classification

Consider a text classification task with $K$ classes. In the text representation stage, the label embedding work is fused [13], and the text and label features are learned in the same latent space, which can inhibit the confusing text from deviating from the golden label. The input text can be expressed as:

$$s_i = [CLS], L_1, L_2, ..., L_K, [SEP], e_1, e_2, ..., e_s, [SEP]$$  \hspace{1cm} (1)
As shown in Fig. 3, use the encoder to obtain the feature representation $h_i$ of the text $s_i$ and applies the softmax function to calculate the predicted function calculates the predicted probability $\hat{y}_i$ of each category, and $\sum_{j=1}^{K} \hat{y}_i = 1$ formalized by:

$$x_i = Wh_i + b$$

(2)

$$\hat{y}_i = \text{softmax}(x_i)$$

(3)

Where $W$ and $b$ are the learnable weight matrix and bias respectively. The maximum value in the probability $\hat{y}_i$ is the category prediction value of the current input text $s_i$. In classification tasks, we usually use cross-entropy (CE) loss. For a dataset $\{x_i, y_i\}_{i=1,...,N}$ containing $N$ samples, define the following cross-entropy loss:

$$L_{ce} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_i^j \log(x_i^j)$$

(4)

Although CE loss works well in most cases, since the input sample label is represented as a one-hot vector. CE loss is not sensitive to obfuscated text, so it cannot effectively deal with it in the dataset.

![Diagram of classifier](image)

**Fig. 3.** Classifier

### B. Screening Confusing Text and Straightforward Text

The input text $s_i$ is subjected to standard classification to obtain the probability predictions of different classes. Then a filter function is designed to filter confusing text and straightforward text. For Chinese confusing text, as it is difficult for the classifier to learn the personality characteristics of the correct class from the current text, the similarity to the feature vector of the confused category is high, and the prediction scores of the two classes are very close. The straightforward text can quickly converge during model training, and the error between the predicted and ground-truth class is small. This kind of text is also defined as a easy sample, and ordinary text is other texts. From the perspective of the loss function, the loss of easily confusing text is relatively large during training, and the loss of straightforward text is relatively small. To this end, we introduce a screening strategy for these two texts.

The text whose difference between the first two categories of predicted probabilities is within threshold $\lambda$ will be judged as confusing text. At this time, the text representation of this text is screened out, and the formalization is as follows:

$$\lambda \geq \max(\hat{y}_i) - \max(\hat{y}_i - \max(\hat{y}_i))$$

(5)

The screening rules for straightforward text are as follows:

$$\max(\hat{y}_i) \geq \sum_{j=1,j\neq \max(\hat{y}_i)}^{K} \hat{y}_i$$

(6)

Straightforward text is essential in shrinking confusing text to the correct class, more effectively comparing the feature similarity of two type of text, and improving the model’s ability to classify confusing text.

### C. Training Strategy based on Triplet Network

For the Chinese confusing text training strategy, first randomly select positive and negative samples from straightforward texts, construct a triplet sample group by combining the two texts with the confusing text, and add a negative sample randomly selected from ordinary text. The triplet loss function is obtained through the triplet network to make the training target closer to the distance between the anchor sample and the positive example while keeping the anchor sample away from the negative example. The modified triplet loss is designed as follows:

$$L_{ct} = \sum_{i,j,k}^{N} [d(x_i,x_j) - d(x_i,x_k) + \delta_1]_+$$

$$+(1 - \beta) \sum_{i,j}^{N} [d(x_i,x_j) - d(x_i,x_j) + \delta_2]_+$$

(7)

where $[z]_+ = \max(z,0)$, $x_i$ is confusing text, $x_j$ and $x_k$ are straightforward text, and $x_j$ is ordinary text.

The first item is called the strong push to build from the straightforward texts, and the second is the weak push to add the selection and construction from the ordinary texts to play a balancing role. In order to allow the model to dig out the features in the straightforward text deeply, set the weight of the first item to be greater than the second item, set to 0.2. The threshold $\delta_1$ and $\delta_2$ is a margin that is enforced between positive and negative pairs. $d(x_i,x_j)$ represents the feature similarity distance between samples. We select Euclidean distance as the distance measurement method, and the sample feature vector is mapped to the Euclidean distance space to achieve the goal of triplet learning.

$$L = \alpha L_{ct} + (1 - \alpha) L_{ce}$$

(8)

In the training, we designs and uses an optimized objective function based on the triplet loss function to ensure that the ordinary text is not affected while training the confusing text.
The objective function is shown in Eq. 8. \( \alpha \) is a hyperparameter
Used to adjust the weight of both.

IV. EXPERIMENTS AND ANALYSIS

A. Datasets

We conduct experiments on Chinese text classification
benchmark datasets with various granularities. 1) nlpcc2014 is
derived from the emotion recognition of Weibo comments in the
NLPCC2014 task. It is a dataset with seven classes of emotion
classification tasks. 2) waimai_10k is a two-category sentiment
classification task dataset for takeaway meal evaluation. 3) THUCNews
is based on the historical data of Sina News RSS
subscription channels. After data cleaning, reintegration, and
division into finance, stocks, science, society, politics, and
entertainment, a total of six classes of news subject classification
datasets. 4) SHNews uses the open-source Sohu news dataset for
data cleaning to remove some missing label data in the data. The
data has 12 categories, including entertainment, finance, real
estate, tourism, technology, sports, health, education,
automobiles, news, culture, woman. In the data preprocessing
stage, text data is normalized by removing renumber cards,
special character fragments, and improving the standardization
of the data. The experimental data was partitioned into training,
testing, and validation sets at a ratio of 8:1:1. See Table 2 for
detailed statistics on datasets.

B. Implementation Details

The experimental parameters in this paper mainly include
classifier model parameters, confusing filter hyperparameters,
and triplet loss parameters. The model is trained using the Adam
gradient descent algorithm, the batch number is 128, the
maximum number of rounds is set to 20 rounds, and the initial
learning rate is set to 2E-5. According to the specific
characteristics of different datasets, the parameter sensitivity
analysis of the confusing category threshold \( \lambda \in [0.02 \sim 0.2] \)
of the confusing filter is carried out. The optimal parameter is
finally selected as the experimental parameter.

C. Baseline Methods

In order to evaluate the classification effect of the proposed
method on confusing text, we use BERT [14] as the benchmark
encoder. It selects methods that perform better in hard sample
and confusing text to conduct comparative experiments.

- **EDA**: Through data enhancement on Chinese confusing
text, balance the amount of data to improve the learning
ability of the model for confusing text.

- **Focal-Loss**: Build a hard sample loss function to
alleviate the problem that a small amount of confusing
text and a large amount of ordinary text contribute
differently to classifier learning [15].

- **H-SCL**: Supervised Contrastive Learning for Chinese
Confusing Text, Better Performance than Random Sampling for Unsupervised Contrastive Learning.

- **TN**: Targeted training on obfuscated text using
traditional triplet network methods.

D. Analysis of Experimental Results

The Accuracy of different methods on the four datasets are
shown in Table 3. It can be seen from the experimental results
that the model processed for confusing text is generally better
than the benchmark model BERT, and the accuracy rate has
been improved to a certain extent. It is highly beneficial to
optimize the model's training method and process the text data
while retaining the benchmark encoder. This can mitigate the
adverse effects of confusing text. Among them, the method
proposed in this paper performs better overall on the four
Chinese text classification datasets than the other listed methods.
The accuracy rates on waimai_10k, nlpcc2014, THUCNews, and
SHNews have increased by 1.45 %, 1.18%, 0.95%, and 0.96%,
respectively. Compared with traditional triplet network methods,
the proposed method has improved the accuracy rates by 0.49%,
0.14%, 0.40%, and 0.26% respectively. Although the results are
not particularly impressive, they provide some insights and ideas
for studying the issues with Chinese confusing text. At the same
time, if the label embedding technology (LE) is removed, the
classification performance of the method in this paper will
decline to a certain extent, proving the proposed method's
rationality and effectiveness. To demonstrate the effectiveness
of our method in addressing the issue of text obfuscation and
check its compatibility with various classification models, we
conducted relevant experiments using other pre-trained models
like RoBERTa[16] and XLNET[17], which are indicated in
Table 4. The experimental results show that the proposed
method can still significantly enhance the classification
performance of the replaced BERT when other classification
models are used. Further clustering of confusing text in the data
can improve the classification performance of the model. The
experimental results confirm the necessity of handling text
obfuscation issues in Chinese text classification research.

### Table II. Dataset Statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Classes</th>
<th>Type</th>
<th>Train</th>
<th>Dev.</th>
<th>Test</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlpcc2014</td>
<td>7</td>
<td>Sentiment</td>
<td>13324</td>
<td>2855</td>
<td>2855</td>
<td>50</td>
</tr>
<tr>
<td>waimai_10k</td>
<td>2</td>
<td>Sentiment</td>
<td>8391</td>
<td>1798</td>
<td>1798</td>
<td>40</td>
</tr>
<tr>
<td>THUCNews</td>
<td>6</td>
<td>Topic</td>
<td>48000</td>
<td>6000</td>
<td>6000</td>
<td>20</td>
</tr>
<tr>
<td>SHNews</td>
<td>12</td>
<td>Topic</td>
<td>22699</td>
<td>5755</td>
<td>5764</td>
<td>20</td>
</tr>
</tbody>
</table>

### Table III. Experimental Results

<table>
<thead>
<tr>
<th>Method</th>
<th>nlpcc2014</th>
<th>THUCNews</th>
<th>SHNews</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>90.04</td>
<td>64.57</td>
<td>93.57</td>
</tr>
<tr>
<td>(+) EDA</td>
<td>90.71</td>
<td>64.74</td>
<td>93.03</td>
</tr>
<tr>
<td>(+) Focal-Loss</td>
<td>90.82</td>
<td>65.53</td>
<td>94.25</td>
</tr>
<tr>
<td>(+) OHEM</td>
<td>91.10</td>
<td>65.92</td>
<td>94.32</td>
</tr>
<tr>
<td>(+) H-SCL</td>
<td>90.88</td>
<td>65.49</td>
<td>94.42</td>
</tr>
<tr>
<td>(+) TN</td>
<td>91.00</td>
<td>65.81</td>
<td>94.12</td>
</tr>
<tr>
<td>(+) Ours w/o LE</td>
<td>91.32</td>
<td>65.95</td>
<td>93.87</td>
</tr>
<tr>
<td>(+) Ours</td>
<td>91.49</td>
<td>65.75</td>
<td>94.52</td>
</tr>
</tbody>
</table>

### Table IV. Performance of Our Method on Other Models

<table>
<thead>
<tr>
<th>Method</th>
<th>nlpcc2014</th>
<th>THUCNews</th>
<th>SHNews</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa</td>
<td>90.88</td>
<td>64.71</td>
<td>94.02</td>
</tr>
<tr>
<td>(+) Ours</td>
<td>90.99</td>
<td>65.28</td>
<td>94.18</td>
</tr>
<tr>
<td>XLNet</td>
<td>91.27</td>
<td>64.46</td>
<td>93.62</td>
</tr>
<tr>
<td>(+) Ours</td>
<td>91.43</td>
<td>64.53</td>
<td>93.68</td>
</tr>
</tbody>
</table>
E. Hyperparameter Influence

The experiment was designed to explore the impact of the confusion text filter threshold parameter $\lambda$ on the model performance. The filter threshold determines the level of tolerance for confused text, and the optimal parameter for the filter threshold varies across different datasets due to differences in text characteristics, quality and length. This paper aims to explore the range of data confusion ratios for which the proposed method enhances the performance of the model to a significant extent. The condition for such an analysis is that all other hyperparameters are set to their respective optimal values. The results of our experiments are depicted in Fig. 4. The selection of filter thresholds has a significant impact on the improvement of the model's performance. Our experiments show that the optimal range for the filter threshold selection is between 0.03 and 0.1, where the overall improvement in the model's performance is most significant.

Fig. 4. Accuracy under different confusion ratios

In order to further illustrate the performance improvement of the triplet loss function proposed in this paper compared with the traditional triplet loss function, we selected happiness, like, and surprise of data from the nlpcc2014 test dataset and used the t-SNE (t-distributed Stochastic Neighbor Embedding) visually display the text features, as shown in Fig. 5.

Fig. 5(a) shows the two-dimensional space representation of the feature vectors of the three categories after the benchmark model BERT is trained. The distribution of the test set data in the embedding space is relatively scattered, the problem of text confusion is prominent, and the model's classification performance is restricted. Fig. 5(b) uses the traditional triple loss to build a model for training. Compared with Fig. 5(a), there is a significant difference in the distance between different categories, but the problem of interleaved stacking of happiness and like categories is prominent, and the distance between classes is relatively scattered. The use of triplet loss cannot eliminate the wrong movement of samples but can only constrain and suppress this negative trend, so there are still many outlier samples interlaced with each other in Fig. 5(b) and Fig. 5(c). Fig. 5(c) is the optimized model using the method of this paper. It is obvious that the distance between the same categories has been shortened, and the confusion problem has been significantly improved. The outlier text has also been reduced, which shows that the confusing text has been corrected. Strengthen the model's ability to distinguish sentiment data.

V. CONCLUSION

In this paper, we propose a text classification method based on triplet network to investigate the impact of confusing text on Chinese text classification tasks. Our proposed method offers a simple yet effective way to enhance the overall performance of the model without making any structural changes. The experimental results presented in this paper confirm the effectiveness and rationality of our approach. However, there are also some deficiencies, such as the requirement to fine-tune the filter threshold parameters for different datasets, which reduces flexibility. There is a need to propose targeted optimization methods for other areas of natural language understanding. In our subsequent work, we will endeavor to produce adaptive weights by taking into account both the overall quality and average distance of the dataset, to further optimize the detection of confusing text in various domains.
ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No.62102136), the Key R & D projects in Hubei Province (No.2021BAA188, 2021BAA184, 2022BAA044), the Science and Technology Innovation Program of Hubei Province (No.2020AE008).

REFERENCES


Applications of Machine Learning in Requirements Traceability: A Systematic Mapping Study

Xingfu Li¹, Bangchao Wang¹,²,³, Hongyan Wan¹,², Yang Deng¹, Zihan Wang³
¹School of Computer Science and Artificial Intelligence, Wuhan Textile University, Wuhan, China
²Engineering Research Center of Hubei Province for Clothing Information, Wuhan Textile University, Wuhan, China
³School of Mathematical & Physical Science, Wuhan Textile University, Wuhan, China

lixingfu1999@163.com, wangbc@whu.edu.cn, hywan@wtu.edu.cn, dd0028y@163.com, wanzh011031@163.com

Abstract—Requirements traceability (RT) is crucial for requirement management and impact analysis of requirement change in software development. The applications of machine learning (ML) technologies to RT have received much attention. In this paper, we aim to provide the state-of-the-art progress of the studies on the intersection of ML and RT. A systematic mapping study (SMS) is conducted and 26 studies have been identified as primary studies. The results present 32 ML technologies and 7 enhancement strategies for establishing trace links. Besides, 46 datasets are utilized for validating the performance of these ML technologies. Additionally, the overall quality of these primary studies is at a good level. This study indicates that numerous studies have proved the potential of utilizing ML technologies for predicting emerging trace links in RT by utilizing existing traceability information. Moreover, open-source datasets are the most popular, which greatly improves the reproducibility of studies. However, there is still a gap between academia and industrial application because of the lack of industrial practice and guidance from practitioners.

Keywords—requirements traceability, machine learning, quality assessment, systematic mapping study.

I. INTRODUCTION

Requirements traceability (RT) is “the ability to describe and follow the life of a requirement, in both a forwards and backwards direction” [1]. It is one of the vital activities in Requirements engineering (RE). It is beneficial for software development that RT helps to identify the origin of requirements, analyze the impact of requirements change and trace the relationships between stakeholders and other artifacts [2]. This ensures the needs of stakeholders are always met in development process, and promotes the transparency and traceability of the software development process.

There are various technologies such as information retrieval (IR) and machine learning (ML) approaches. However, it cannot use historical information for prediction, which leads to poor universality. Especially some informal products still require a lot of manual intervention. After 2017, increasing ML technologies are developed and applied to RT to obtain complete requirements trace links. It can learn from existing traceability information to obtain characteristics. The applications of ML in RT approaches have been emerging as a hot topic with the efforts of researchers and practitioners. However, there has been no systematic literature review about the status of ML-based RT approaches over the past ten years. It prompts us to summarize the progress of ML-based RT.

In this paper, we provide evidence-based insights of the intersection of ML and RT. We hope that it can help researchers and practitioners better understand ML-based tracing approaches and extend them through the novel study.

This paper is structured into six main sections: Section I provides an introduction to the background. Section II introduces the related work and research questions (RQs). Section III and Section IV provide the execution process of the SMS method and the findings, respectively. Section V discusses the validity threat and potential research directions. At last, the conclusion is summarized in Section VI.

II. RELATED WORK

As far as we know, there are some relevant reviews about RT. Wang et al. provide a review of the intersection of RT and IR. They have summarized 21 enhancement strategies that can improve the performance of four kinds of IR technologies [3]. Torkar et al. have concluded available tools and galore techniques for RT [4]. However, they don’t pay close attention to the state-of-the-art progress of ML-based RT approaches. Tufail et al. have performed a systematic review and identified seven kinds of RT models, ten challenges, and fourteen tools [2]. However, they don’t summarize the specific technologies to guide researchers.

Wang et al. have conducted a systematic literature review and discussed challenges associated with RT activities [5]. They also summarize RT technologies that include ML technologies and evaluate the overall quality of primary studies. It is noted that the time interval of their work is 2006–2016, which indicates that more novel technologies haven’t been covered. The ML-based SE approaches have been reviewed by Mezouar et al. [6]. However, their work doesn’t focus on the RT field.

However, there has been no review about the application of ML in RT field in the past decade. It is desired to comprehensively analyze these ML techniques to understand emerging study directions in RT. The aim of this paper is to systematically investigate ML-based tracing approaches over the last decade. Besides, the overall degree of quality is quantified to provide a reference for researchers. In summary, this study has addressed four RQs to provide the current progress of the studies on the intersection of ML and RT:

RQ1: What are the publication times and venues of primary studies?
RQ2: Which ML approaches and strategies are applied to RT?
RQ3: Which datasets are utilized for ML-based RT approaches?
RQ4: What is the overall quality of primary studies?

III. RESEARCH METHOD

This SMS has followed the guidance provided by [7]. In this section, a detailed search strategy, classification process and quality assessment method are defined to conduct SMS.
A. Search Strategy

In this section, the search terms and databases are described for searching. Besides, selection criteria are proposed to screen primary studies. The whole search process is shown in Fig. 1.

1) Search Scope and Terms

Five common databases are selected as follows: ACM Digital Library, IEEE Xplorer, Science Direct, SpringerLink, and EI Compendex. These databases cover a wide range of studies related to ML and RT. In addition, this SMS is conducted based on relevant literature from the past ten years, with a search period from January 2013 to December 2022.

According to the four RQs of this study, the based search terms are defined as "requirements traceability" and "machine learning". In order to refine these search terms, alternative and related terms in the RT field are used to iteratively retrieve 5 databases according to the PICO standard [3]. The finalized search terms are as follows:

- **Population**: requirements traceability, requirements trace, requirements tracing, requirements traceability recovery.
- **Intervention**: machine learning, ML, supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning.

After determining search terms, logical operators (i.e., OR and AND) can be used to connect strings to build more complex search queries. The search string is formulated as follows: "requirements traceability" OR "requirements trace" OR "requirements tracing" OR "requirements traceability recovery" AND "machine learning" OR "ML" OR "supervised learning" OR "unsupervised learning" OR "semi-supervised learning" OR "reinforcement learning".

2) Study selection criteria

In this step, inclusion and exclusion criteria are designed to select primary studies that are related to this research. The following are the inclusion and exclusion selection criteria:

- **Inclusion selection criteria**:
  1: The study is published from January 2013 to December 2022.
  2: The study is written in English.
  3: The study with the more detailed description is selected when multiple authors have the same study.

- **Exclusion selection criteria**:
  1: The study is a review or gray literature.
  2: The study with less than 3 pages.
  3: Duplicate studies with the same authors.

3) Study selection procedure

The Zotero tool is used to perform the search and selection process. Firstly, we utilize search strings constructed in III.A to retrieve 910 literatures from five digital libraries. Among them, EI Compendex contains nearly one-third of literature (302/910). Secondly, the Zotero tool is employed to remove duplicates resulting in 713 unique studies. Next, inclusion and exclusion criteria (I1-I2, E1-E3) are applied to perform a coarse-grained filter on these 713 studies. Subsequently, I3 and E4 criteria are employed to exclude irrelevant literatures based on their titles, abstract and keywords. A total of 221 relevant papers are obtained after this round of filtering. Finally, 25 literatures are selected by reading the full text. In addition, snowballing is performed to prevent the omission of related studies. Ultimately, 26 literatures are strictly selected as the primary studies.

B. Data Classification and Analysis

After determining 26 primary studies, the data items that need to be extracted from each primary study have been confirmed by discussing according to our goals. Afterwards, the first two authors extracted data from all the primary studies separately. We have compared these two extraction reports to identify controversial points. Finally, all authors have discussed and determined the final results through the seminar. The extraction report for consultation is generated in a Word file. Additionally, the technical report and detailed search process are available on the website (https://github.com/WTU-intelligent-software-development/ML-based-RT-SMS).

C. Quality Assessment

During the quality assessment phase, the quality of all primary studies has been evaluated. Table I presents that these primary studies are rigorously assessed based on four dimensions selected according to the technology transfer model [8]. Each evaluation dimension is divided into different levels of evidence for precise quantification, and Table I provides a detailed description of these evidence levels and their corresponding scores. Then, the scores obtained from these evaluations are utilized to conduct a comprehensive analysis to reflect the extent to which these studies support technology transfer.
The first evaluation dimension used for quality assessment (Research Method) is proposed by Alves et al. [9]. It is one of the key factors to ensure the credibility and effectiveness of research results. The correct selection and appropriate interpretation of research methods are one of the key factors to ensure the credibility and effectiveness of research results.

Context and Subjects proposed by Ivarsson et al. [10] are adopted as the second and third dimensions. Context includes industrial context and academic context. Validation in the academic environment provides support for industrial applications. The subject usually includes different roles such as students, researchers, and practitioners. Various subjects represent different research abilities and levels of experience.

In addition, providing a dataset source can increase the credibility of study and provide more opportunities for other researchers to expand their research. Therefore, the source of the dataset has been added as the final evaluation dimension. Finally, the evaluation method proposed by Wang et al. [5] has been utilized to assign scores to each evidence level.

IV. RESULTS

A. RQ1: What are the publication times and venues of primary studies?

Fig. 2 shows the number of studies published per year in this field and the distribution of venues from 2013 to 2022. A total of 26 primary studies are published, covering 7 types of journals and 10 types of conferences. We can see that studies have become plentiful since 2017. Requirement Engineering Conference (RE Conference) and International Conference on Software Engineering (ICSE) are the venues with the most publications. It should be noted that the underlined venues are conferences and their names are abbreviations due to the space limit. The full names of venues are recorded in the technical reports if readers are interested.

B. RQ2: Which ML approaches and strategies are applied to RT?

The illustration of the general process of ML-based RT approaches is shown in Fig. 3. It is usually divided into three stages: preprocessing stage, link generation stage, and link refinement stage. The preprocessing stage includes the data preprocessing process and the generation of the feature vector process. The main task of the link generation stage is to filter out candidate links. The link refinement stage optimizes the candidate links generated in the link generation stage.

C. RQ3: Which datasets are utilized for ML-based RT approaches?

The datasets used in each primary study are summarized to analyze datasets used to verify ML-based RT approaches. Table III shows the specific information, source links, frequency of use, and primary studies of these datasets. There are 46 datasets that have been used 100 times in total. Datasets that have a source link and are used more than four times are chosen to display because of space limitations. If you need more detailed information, please see the technical report.
Fig. 3. The General Process of ML-based Requirements Traceability

<table>
<thead>
<tr>
<th>ML Technologies</th>
<th>Strategies</th>
<th>Applying Stage</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>Link Classification [11][13][14][15][16][17][18][19]</td>
<td>G</td>
<td>8</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>Link Classification [11][13][14][15][16][17][19][20][21][22]</td>
<td>G</td>
<td>8</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>Link Classification [13][14][16][17][19][20][21][22]</td>
<td>G</td>
<td>8</td>
</tr>
<tr>
<td>Word2vec</td>
<td>Representation Learning [12][23][24][25][26]</td>
<td>P</td>
<td>5</td>
</tr>
<tr>
<td>Feedforward Neural Network (FNN)</td>
<td>Semantically Similar Words Extraction[24], Link Classification [29]</td>
<td>P</td>
<td>2</td>
</tr>
<tr>
<td>Gradient Boosting Decision Tree</td>
<td>Link Classification [11][20]</td>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>Link Classification [13][16]</td>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>Hierarchical Agglomerative Clustering (HAC)</td>
<td>Semantically Similar Words Extraction[30][31]</td>
<td>P</td>
<td>2</td>
</tr>
<tr>
<td>Recurrent Neural Networks (RNN)</td>
<td>Representation Learning [26], Link Classification [29]</td>
<td>P</td>
<td>2</td>
</tr>
<tr>
<td>Active Learning (AL)</td>
<td>Link Classification [18][32]</td>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>Bagging</td>
<td>Link Classification [16][21]</td>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>Bi-directional Encoder Representation from Transformers (Bert)</td>
<td>Representation Learning [26]</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>Bi-directional Gated Recurrent Unit (Bi-GRU)</td>
<td>Representation Learning [26]</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>Fasttext</td>
<td>Representation Learning [27]</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>Global Vectors (GloVe)</td>
<td>Representation Learning [23]</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>K-medoids</td>
<td>Semantically Similar Words Extraction [31]</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>Single Link Clustering</td>
<td>Automatically Build Ontology [28]</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>Bi-directional Long Short-Term Memory (Bi-LSTM)</td>
<td>Representation Learning [26]</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>Gated Recurrent Unit (GRU)</td>
<td>Representation Learning [26]</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>Universal Sentence Encoder (USE)</td>
<td>Representation Learning [27]</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>K-means</td>
<td>Documents Clustering [33]</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>Logit Boost</td>
<td>Link Classification [21]</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>Hierarchical Bayesian Network</td>
<td>Link Classification [34]</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>Label Spreading</td>
<td>Link Classification [23]</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>Ranking SVM</td>
<td>Learn to Rank [12]</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>RankBoost</td>
<td>Learn to Rank [29]</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>Reinforcement Learning (RL)</td>
<td>Link Classification [35]</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>Spectral Clustering</td>
<td>Graph Clustering [36]</td>
<td>R</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: “P” represents Preprocessing Stage, “G” represents Links Generation Stage, “R” represents Links Refinement Stage.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Source Artifacts</th>
<th>Target Artifacts</th>
<th>True link</th>
<th>Source link</th>
<th>Freq.</th>
<th>Primary studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTour</td>
<td>Use Case</td>
<td>Code</td>
<td>366</td>
<td>12 [11][12][13][14][15][18]</td>
<td>[19][20][21][25][32][34]</td>
<td></td>
</tr>
<tr>
<td>EasyClinic</td>
<td>Use Case</td>
<td>Code</td>
<td>93</td>
<td>9 [12][13][14][18][19]</td>
<td>[21][23][32][36]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interaction Diagram</td>
<td>26</td>
<td>9 [12][13][14][18][19]</td>
<td>[21][23][32][36]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use Case</td>
<td>Use Case</td>
<td>53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test Case</td>
<td>Use Case</td>
<td>93</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iTrust</td>
<td>Use Case</td>
<td>Code</td>
<td>534</td>
<td>7 [12][14][19][24]</td>
<td>[25][35][36]</td>
<td></td>
</tr>
<tr>
<td>SMOS</td>
<td>Use Case</td>
<td>Code</td>
<td>1045</td>
<td>8 [11][13][14][15][18]</td>
<td>[19][22][32][34]</td>
<td></td>
</tr>
<tr>
<td>CM-1</td>
<td>High-level requirement</td>
<td>Low-level design document</td>
<td>Unclear</td>
<td>7 [12][14][19][24]</td>
<td>[25][35][36]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High-level requirement</td>
<td>Low-level requirement</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Requirement</td>
<td>Design</td>
<td>46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Requirement</td>
<td>Use Case</td>
<td>Unclear</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eAnci</td>
<td>Use Case</td>
<td>Code</td>
<td>567</td>
<td>7 [11][13][14][15][18][19][32]</td>
<td>7 [12][18][24][32]</td>
<td></td>
</tr>
<tr>
<td>MODIS</td>
<td>High-level requirement</td>
<td>Low-level requirement</td>
<td>41</td>
<td><a href="http://promisest.ualberta.ca/SERepository">http://promisest.ualberta.ca/SERepository</a></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Note1: There are 23 open-source datasets with only one usage frequency, i.e., Albergerate, GANNNT, CCHIT, EBT, LibEST, Select SI, Chess, Gantt, JHotDraw, ARC-IT, DASHBUILDER, JBTM, Accumulo, Ignite, Isis, Tika, Care2x, CREST, Physician, Trial Implementations, PatientOS, PracticeOne, WorldVistA.

Note2: There are 16 datasets without the source link, i.e., Pine, Drools, Lucene, PTC, Waterloo, AIRFLOW, ANY23, IMMUTANT, CAF and 7 unnamed datasets
mentioned in Section III.B.

Nearly 65% (30/46) of datasets that are used in primary studies are open-source datasets. These open-source datasets have been used 81 times (81%, 81/100). Besides, less than 27% (7/26) of primary studies don’t fully use open-source datasets. More importantly, the majority (59/81) of all open-source datasets are provided by CoEST. Moreover, eTour, EasyClinic, and iTrust are the three most popular open-source datasets.

D. RQ4: What is the overall quality of primary studies?

Fig. 4 has summarized the overall quality of each assessment dimension. Fig. 5 uses box diagrams to analyze the overall quality of all primary studies comprehensively. The following information could be obtained by analyzing the data listed in Fig. 4 and Fig. 5:

a) The vast majority of studies (around 85%) use the experiment research method, while only 15% (4/26) of primary studies use case study research method. It indicates that most studies lack validation in real industrial environments, which hinders researchers from understanding more complex phenomena.

b) More than half of primary studies (17/26, around 65%) are validated in the academic context. According to the technology transfer model [8], these studies have a large gap from actual industrial applications, and industrial validation (such as static and dynamic validation) is still needed before the final solution can be released.

c) Only two primary studies subjects are practitioners. It indicates that most studies lack guidance from practitioners’ practical and professional industry knowledge. Therefore, it is suggested that more practitioners can participate in evaluation and validation.

![Fig. 4. The Distribution of Quality Score of Each Assessment Dimensions](image)

![Fig. 5. Overall Quality Assessment Scores of All Primary Studies](image)

d) Half of the studies don’t provide a dataset source. The lack of datasets source not only affects reproducibility but also reduces the transparency and credibility of the study. It is noted that the four studies [11][19][24][30] do not provide dataset sources because of confidentiality.

We have calculated the sum of the quality scores for each primary study. It should be noted that each quality assessment criterion has a maximum score of 1, and the full quality score for each study is 4. The quality levels are divided into four categories (poor: 0-1.99, middle: 2.0-2.59, good: 2.6-3.19, excellent: 3.2-4) [5][10]. From Fig. 5, the quality scores are mainly concentrated between 2.6 and 3.1. Moreover, the median and mean scores suggest that the overall quality of primary studies is at a "good" level. This proves that this SMS is trustworthy.

V. DISCUSSION

A. Validity Threats Discussion

The threats that influence processes and the findings of this SMS are introduced in this section. They are divided into internal validity, external validity, conclusion validity, and construction validity. Four threats are introduced as follows:

Internal validity: To avoid the risk of internal threats, data extraction and classification should be carefully conducted to ensure their accuracy. Therefore, in this study, the first two authors collaborated closely and conducted repeated reviews to minimize potential errors.

External validity: External validity concerns the generality of the conclusions of our SMS. For instance, whether the primary studies can represent the research questions of the review topic. In order to mitigate the threat, multiple search processes have been conducted and the search terms were constantly adjusted to improve the coverage.

Conclusion validity: The literature may be excluded wrongly. Therefore, the first two authors simultaneously conducted the screening process and determined the final primary study through discussion among the entire authors.

Construction validity: Construction validity is concerned with the deviation of the studied concept from the studied topic. The based search terms are "requirements traceability" and "machine learning" To reduce the impact, synonyms or alternative terms are used for searching to maximize coverage.

B. Future Research Directions

This SMS has illuminated several future research directions with great potential for further exploration in RT.

Firstly, the mainstream ML models applied to RT are traditional models such as Random Forest, Decision Tree, and Naïve Bayes at present. With the development of ML or deep learning, the applications of novel technologies to RT are a potential future research direction.

Secondly, due to the ability of different technologies to handle different features, mining a combination of multiple machine learning techniques may improve the accuracy of feature based link recognition.

Finally, the utilization of many types of features may lead to feature redundancy and even have a counterproductive effect. Exploring a suitable feature selection approach to improve the performance of models is worth study.

VI. CONCLUSION

This SMS presents that ML techniques are playing a role in RT and researchers pay more attention to applying ML to
RT to obtain accurate and complete trace links. The following conclusions can be drawn:

1) Seven enhancement strategies have been conducted to support the establishment of trace links. ML models can learn from existing traceability information to predict new links. Moreover, link classification has attracted much attention to distinguish whether artifacts have trace links.

2) Open-source datasets are more popular than closed-source datasets. Researcher CoEST is the most popular source of datasets. In order to make the study more reproducible, researchers are suggested to use open-source datasets.

3) The overall quality is at a good level. This indicates that the research level in this field is relatively good, and the selected literature is representative and reliable. However, it is also noticed that the research method lowers the overall quality. It is suggested that researchers can verify and evaluate in the realistic industrial environment.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China Project (No. 62102291), and the Opening Foundation of Engineering Research Center of Hubei Province for Clothing Information (No. 2022HBCI02, No. 2022HBCI05).

REFERENCES


FeDeFo: A Personalized Federated Deep Forest Framework for Alzheimer’s Disease Diagnosis

Chen Qian*, Hui Xiong† and Jiyun Li‡
School of Computer Science and Technology, Donghua University
Shanghai, China
Email: *chen.qian@dhu.edu.cn, †xionghui0310@foxmail.com, ‡jyli@dhu.edu.cn

Abstract—Alzheimer’s disease (AD) is a neurodegenerative disease that severely affects cognition, memory, and behavior and is incurable. Mild cognitive impairment (MCI) is a clinical precursor to AD, and early diagnosis of AD is essential for the prevention and intervention of disease progression. The hippocampus is one of the first brain regions affected by AD, and therefore structural magnetic resonance images (sMRI) are commonly used to measure the shape and volume of the hippocampus. In this paper, we propose a federal deep forest model called FeDeFo for calculating hippocampal volume using sMRI images to achieve AD classification. Firstly, to effectively protect data privacy, we use a federated learning framework to collaboratively train a gradient boosting decision tree (GBDT) model based on the local data of each client. In addition, to address the data discrepancy between clients, we introduce a deep forest model to exploit the local data beyond local interactions further and fuse it with the federally trained GBDT to personalize the model for each client. The experiments demonstrate that our proposed approach is able to personalize the model while protecting the data privacy of each client, providing a new idea for AD classification.

Keywords—federated deep forest; personalized federated learning; Alzheimer’s disease diagnosis.

I. INTRODUCTION

Alzheimer’s disease (AD) is a fatal neurodegenerative disorder with insidious onset in the presenile period [1]. It affects memory and cognitive ability in an irreversible manner and gradually causes the decline of the quality of daily life and social functions. Although developed AD has no cure, it can be delayed or even prevented at the earliest stage, known as mild cognitive impairment (MCI) [2]. Thus, distinguishing MCI from AD or normal cognition (NC) is an urgent need in AD diagnosis. So far, neuroimaging is the best non-invasive technique to look for abnormalities in the human brain [3].

Undoubtedly, deep learning has revolutionized image processing. It can solve difficult problems such as image colorization, classification, segmentation, and detection [4]. For example, a deep neural network (DNN) is a stack of multiple layers, which allows models to become more efficient at learning complex features and performing more intensive computational tasks. It outshines classic machine learning paradigms in machine perception tasks involving unstructured data. However, DNN has a number of well-known limitations, e.g., living off a considerable amount of data, requiring enormous computational resources, and lacking theoretical explanations [5]. For the above reasons, researchers are looking for an alternative paradigm. Eventually, deep ensemble learning gets attention, such as GrowNet [6], S-DNN [7], DSN [8], deep forest [9], because it combines the advantages of both the DNNs as well as ensemble learning such that the final model has better generalization performance [10].

However, there is another challenge in the healthcare area, which is the well-known privacy problem [11]. Traditional deep learning generally proceeds in two phases: collecting data from different participants for preprocessing and feeding the data to a monolithic model for training. As a result, the risk of privacy leakage is unavoidable. To tackle the problem, we present a federated learning framework. It enables distributed clients to collaboratively train a shared model without sharing their training data. To be specific, model parameters are computed locally by each client device and exchanged with a central server, which aggregates the local models for a global view [12]. It is worth noting that our framework provides personalization in order to improve the privacy-accuracy tradeoffs and balance the benefits among different parties [13].

II. RELATED WORK

A. Federated Learning

The concept of federated learning was first proposed by Google AI [14], which enables mobile terminals to jointly train a global machine learning model in a decentralized manner without sharing individual data. Thereby, federated learning is an interdisciplinary domain of machine learning and privacy computing. In terms of training samples, it can be broadly divided into two categories: horizontal federated learning and vertical federated learning [12]. The former refers to datasets owned by different parties that share the same feature space but differ in samples, whereas the latter refers to datasets owned by different parties that differ not only in samples but also in feature space.

In the healthcare domain, horizontal federated learning is more frequently used as clinical task demands increase [15]. For instance, PRCL coordinates multiple medical institutions and cloud servers to develop an electronic health records (EHR) system since the hospitals run a neural network using their own records, and the cloud server aggregates the update parameters [16]. Likewise, horizontal federated learning has...
been used in breast cancer prediction [17], blood pressure estimation [18], skin disease detection [19], etc. Recently, experts successfully built a powerful model for COVID-19 screening by federated training using chest X-ray images from different hospitals [20]. In this paper, the proposed FeDeFo framework also adopts horizontal federated learning.

Regarding application scenarios, the types of federated learning mainly include cross-device and cross-silo federated learning [21]. The former usually requires massive mobile devices as trainers, each owning a small amount of raw data, while the latter involves few reliable organizations, each holding medium to large datasets. In this paper, we study cross-silo federated learning due to its suitability for healthcare scenarios [22]. Thereby, our work does not need to pay attention to the scheduling problem and communication bottleneck. In addition, we can assume that the central server is an honest and reliable third party in the training process.

General federated learning approaches face several fundamental challenges. One of them is that the global model can only capture the statistical characteristics of different parties rather than the unique personal styles. For example, differences in age-adjusted Alzheimer’s dementia prevalence exist among regions of the world due to the combination of low average educational attainment and high vascular risk profile [23]. Another is the heterogeneous computing resources and network conditions of different federated learning devices. Fortunately, both challenges can be markedly alleviated by personalized federated learning [24]. Furthermore, since medical data are highly sensitive and valuable, personalized federated learning can even more amplify the model quality. For example, FedHome put forward an edge cloud federated learning architecture for home healthcare services, which allows a client to train a personalized model by the global model and its private data [25]. In this paper, the proposed FeDeFo framework also adopts personalized federated learning in order to heighten the patient experience.

B. Ensemble Learning

Ensemble learning is not a machine learning algorithm but a technical framework that aggregates outputs of multiple models in order to improve the overall performance and generalization. In general, there are three primary ensemble techniques, viz., bagging, boosting, and stacking [10].

Boosting is an influential ensemble methodology referring to a family of algorithms that convert a cluster of weak learners into a strong one. Unlike bagging, boosting can reduce bias by learning in a sequence that iteratively adjusts the weight of observation as per the last classification [26]. Gradient boosting decision tree (GBDT) is a classic additive model that uses a boosting ensemble of decision trees to predict a target label [27]. More specifically, it uses a forward distribution algorithm to perform greedy learning. In each iteration of learning, a classification and regression tree (CART) is used to fit the residuals of the previous one.

Random forest is an enhanced version of the decision trees, which uses the bagging strategy to build multiple decision trees and aggregate them for an accurate result with as little bias as possible [28]. As an evolution of random forest, gcForest achieved an innovation breakthrough by constructing a multi-layer neural network in which a number of random forests, instead of neurons, are embedded in each layer [9]. Overall, gcForest becomes more attractive due to the simple algorithm, fewer initial hyperparameters, and thorough extraction of feature relations. Furthermore, it can ensure higher prediction accuracy with a smaller dataset, adapt to different situations by automatically settling its complexity, and moderate the overfitting issue through its robustness. At present, gcForest and its extensions have been increasingly utilized in the real world. For example, a revised gcForest, namely BCDForest, is proposed to classify cancer subtypes based on small biological datasets [29]. Another one can successfully identify ADHD and control subjects [30].

It is worth noting that any type of classifier is applicable in gcForest, such as GBDT. Additionally, GBDT re-weights the original training sample in every boosting step, so it has an excellent generalization ability suitable for solving regression problems similar to disease diagnosis. Therefore, we tailor-make a deep ensemble algorithm by integrating GBDT into gcForest to diagnose Alzheimer’s dementia.

III. FeDeFo Framework

A. Overview

This paper proposes a novel and practical framework called personalized federated deep forest framework (abbreviated as FeDeFo), which aims to classify subjects with AD, MCI, and NC while preserving privacy. In FeDeFo, multiple clients have their own data that share the same features. Thus, FeDeFo focuses on scenarios suitable for horizontal federated learning. In addition, FeDeFo supports personalization, so each client can have a slightly different local model and hence supply a better customer experience.

We are given data from \( N \) different clients, which are denoted by \( \{C_1, C_2, \ldots, C_N\} \), while the data they provide are denoted by \( \{D_1, D_2, \ldots, D_N\} \). Conventional methods train a model \( M_{ALL} \) by combining all the data \( D = D_1 \cup D_2 \cup \cdots \cup D_N \). All the data have different distributions. In our problem, we want to collaborate all the data to train a federated model \( M_{FED} \), where any client \( C_i \) does not expose its data \( D_i \) to each other. If we denote the accuracy as \( A \), then the objective of our model is to ensure the accuracy of federated learning is close to that of conventional learning denoted by:

\[
|A_{FED} - A_{ALL}| < \Delta,
\]

where \( \Delta \) is an extremely small non-negative real number.

The FeDeFo framework aims to achieve accurate AD classification tasks through federated learning and deep forests without compromising privacy security. There are two participants in FeDeFo: the central server and the client. Every client keeps its patients’ data safe by preventing any other participants from accessing it. In this paper, we choose the GBDT model for local training. As shown in Figure 1,
each training iteration, the central server broadcasts the model to every client. Then, each client trains its own GBDT model using its data and uploads the gradients to the central server. Finally, the server aggregates the gradients in order to upgrade the global model. After the federated training stops, every client weaves the resulting model to the local deep forest model, i.e., gcForest, aiming to train a personalized model and further provide a more accurate AD diagnosis system. To summarize, the FeDeFo framework overcomes information segregation through higher-order information scattered across different clients without exchanging their privacy data.

B. GBDT

GBDT is an ensemble model which trains a sequence of decision trees. Formally, a dataset $D$ with $n$ instances and $d$ features can be described as $D \{ (x_i, y_i) \} (|D| = n, x_i \in \mathbb{R}^d, y_i \in \mathbb{R})$. So, the output can be predicted via $K$-additive functions as follows:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \ f_k \in \mathcal{F},$$

where $\mathcal{F} = \{ f(x) = \omega_q(x) \} \ (q : \mathbb{R}^d \rightarrow T, \omega \in \mathbb{R}^T)$ is the space of regression trees. Here $q$ denotes the structure of each tree that maps an instance to the corresponding leaf index. $T$ is the number of leaves in the tree. Each $f_k$ corresponds to an independent tree structure $q$ and leaf weights $\omega$. Moreover, we use $\omega_i$ to represent the score on the $i$-th leaf. For a given example, we use the decision rules in the trees (given by $q$) to classify it into the leaves and perform the final prediction by summing up the score in the corresponding leaves (given by $\omega$). To learn the set of functions used in the model, we minimize the following regularized objective as follows:

$$\mathcal{L} = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k),$$

where $l$ is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target $y_i$. We hereby define $\Omega(f_k) = \gamma T_l + \frac{1}{2} \lambda \| \omega \|^2$ as a regularization term to penalize the complexity of the model, in which $\gamma$ and $\lambda$ are hyperparameters. Hence, GBDT minimizes the following objective function at the $t$-th iteration as follows:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_t f_t^2(x_i)] + \Omega(f_t),$$

where $g_i = \partial_{y} l(y_i, \hat{y}_i^{(t-1)})$ and $h_i = \partial_{y}^2 l(y_i, \hat{y}_i^{(t-1)})$ are the first-order and second-order gradient statistics on the loss function.

C. Federated Training

Structurally speaking, GBDT is a collection of decision trees constructed in a serial manner. However, it is generally implemented and trained as an integral whole under a standalone environment. Thereby, in FeDeFo, GBDTs are revised to coincide with the decentralized environment. As Figure 2 shows, the process of updating the gradients is modified with the aim of receiving updates from the external.

For each iteration of federated training, each client starts with initializing a local GBDT model at first and feeds the model with private data subsequently. In the GBDT, each decision tree is constructed by fitting the negative gradient of the previous one. Hence, the complete GBDT is shaped as a chain of decision trees. When a client finishes the local training, it calculates the first-order and second-order gradient statistics of the loss function. Then, both gradients are uploaded to the server for aggregation. Eventually, the central server collects all clients’ feedback and updates the global GBDT model. Notably, the server will continue to broadcast the aggregated gradients to every client. Therefore, in the next iteration, those gradients become the building blocks of the new local GBDT. The aforementioned procedure is repeated until the global model has converged and the federated learning result is satisfactory.

D. Personalized Training

Federated learning breakthroughs the data silos by collaborative model training with decentralized datasets. However,
its popularity is not growing in practical terms because the global model usually performs unsatisfactorily on participant-specific data. On this account, the FeDeFo framework provides a personalized training method allowing the model to learn fine-grained information from the ad-hoc client as well as the coarse-grained features from all participants. Inspired by gcForest, we hereby design a personalization model using multi-grained scanning and cascading forests with a view to achieving high performance on feature representation learning with high-dimensional data in the context of AD classification.

As Figure 3 illustrates, the personalization model takes the client’s private data as input. Then, the raw features of input are extracted and processed by a multi-grained scanner to generate feature vectors, which are further sent to the cascade forest to complete the classification task. Notably, the original cascade forest in gcForest is defined by multi-level integration of decision trees, which are theoretically replaceable by any other classifiers that can output class distribution vectors. Here, we adopt GBDTs instead of decision trees to construct the personalization model. In detail, each cascade level contains four GBDTs, each of which outputs a class distribution vector. Subsequently, the four vectors at the same level and the output of the multi-grained scanner are concatenated. The result becomes the input vector of the next level. Finally, the classification result is the maximum of the average class vectors outputted from the last level.

IV. EXPERIMENT
A. Dataset
The data used in this paper are extracted from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) [31] database. As demonstrated in Table I, a total of 400 subjects are involved, including 93 AD patients, 201 MCI patients, and another 106 normals as the control group. Because the hippocampal atrophy can evidently suggest the AD diagnosis [32], we select their sMRI with magnetization-prepared rapid gradient-echo (MP-RAGE). Such an image can offer a clear vision of the hippocampus, which is advantageous for volume estimation.

<table>
<thead>
<tr>
<th>Type</th>
<th>Amount</th>
<th>Age</th>
<th>Sex (Male/Female)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>93</td>
<td>75.15±8.14</td>
<td>50 / 43</td>
</tr>
<tr>
<td>MCI</td>
<td>201</td>
<td>73.61±6.92</td>
<td>133 / 68</td>
</tr>
<tr>
<td>NC</td>
<td>106</td>
<td>76.16±7.14</td>
<td>54 / 52</td>
</tr>
</tbody>
</table>

B. Experimental Process
The hippocampal volume is crucial to diagnose AD and MCI. In this paper, we classify the patients with AD, MCI, and NC by assessing the relative hippocampal volume, which is calculated by the whole brain volume dividing the absolute hippocampal volume. The relative volume can eliminate the difference in brain volume between different populations. The brain images are initially preprocessed by the FSL tool [33], including intensity inhomogeneity correction, skull removal, intensity normalization, and image cutting. The unified image size is $196 \times 271 \times 181$. After that, we can use the 3DUnet-CBAM model [34] to extract the hippocampal part. Finally, we use IBASPM [35] to label the voxel of images in a neuroanatomical manner and further calculate the volumes automatically. The activities in data processing are briefly expressed in Figure 4.

Subsequently, we input the relative hippocampal volume to the GBDT models deployed on both central server and client devices. During the learning process, 70 percent of the data are used for training, whereas the rest are for model evaluation. Once the federated GBDT is mature, it will be adopted as a classifier and loaded into the deep forest for personalization. Finally, every client can use its own model to diagnose AD and MCI.

C. Experimental Results
We also perform experiments on AD classification using deep learning methods (e.g., LeNet and VGGNet), and machine learning method (e.g., SVM). In comparison with those
experimental results, we prove the outperformance of our FeDeFo framework. Moreover, to demonstrate the effectiveness of the personalization mechanism in FeDeFo, we conduct an extra experiment using federated GBDT only, i.e., an equivalent of FeDeFo without personalized training.

To thoroughly investigate the classification performance, we use three metrics for evaluation: accuracy (ACC), sensitivity (SEN), and specificity (SPE). In clinical practice, accuracy measures how correctly a diagnostic test identifies and excludes a given condition, yet sensitivity evaluates how good the test is at detecting a positive disease, whereas specificity estimates how likely patients without the disease can be correctly ruled out. If we denote $TP$ as true-positive samples, $FP$ as false-positive samples, $TN$ as true-negative samples, and $FN$ as false-negative samples, the equations of the metrics can be defined by:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$
$$\text{Sensitivity} = \frac{TP}{TP + FN}$$
$$\text{Specificity} = \frac{TN}{FP + TN}$$

As shown in Table II, FeDeFo outshines LeNet and VGGNet on every metric. The sensitivity and specificity of classifying MCI and NC are slightly lower than SVM, probably because the sample size is too small. Meanwhile, it improves the average results by 6.1% compared to the federated GBDT. In summary, our FeDeFo framework achieves exceptional performance in classifying subjects with AD, MCI, or NC while preserving patients’ privacy.

<table>
<thead>
<tr>
<th></th>
<th>AD vs. NC (%)</th>
<th>MCI vs. NC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACC</td>
<td>SEN</td>
</tr>
<tr>
<td>LeNet</td>
<td>83.8</td>
<td>73.2</td>
</tr>
<tr>
<td>VGGNet</td>
<td>84.7</td>
<td>77.3</td>
</tr>
<tr>
<td>SVM</td>
<td>83.3</td>
<td>71.4</td>
</tr>
<tr>
<td>Fed. GBDT</td>
<td>86.4</td>
<td>76.5</td>
</tr>
<tr>
<td>FeDeFo</td>
<td><strong>91.6</strong></td>
<td><strong>85.7</strong></td>
</tr>
</tbody>
</table>

V. Conclusion

The FeDeFo framework provides a rational combination of deep learning, ensemble learning, federated learning, and personalization. Together, they suggest a promising solution for privacy-preserving AI services. In this paper, we use FeDeFo to produce a family of AD diagnosis systems. The slight
difference between them makes each system more suitable for its own scenario. Theoretically, the FeDeFo framework is extensible to other medical fields or even more. In the future, we will test our FeDeFo framework on distinct datasets using different training models and personalization techniques in more domains.

ACKNOWLEDGMENTS

This work has been supported by the National Key R&D Program of China under Grant 2019YFE0190500, the Fundamental Research Funds for the Central Universities of Ministry of Education of China (Grant No.2232021D-22), and the Initial Research Funds for Young Teachers of Donghua University.

REFERENCES

A Detection-based Attention Alignment Method for Document-level Neural Machine Translation

Kang Zhong¹  Wu Guo¹  Bin Gu¹  Weitai Zhang²  Chao Lin²
¹NERC-SLIP, University of Science and Technology of China, Hefei, China
²State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, Hefei, China

E-mail: Corresponding author: Wu Guo, guowu@ustc.edu.cn

Abstract

Previous works have shown that inter-sentential contextual information can lead to substantial improvements in document-level neural machine translation (DocNMT). Most existing DocNMT models focus on methods of introducing inter-sentential contextual information through attention mechanisms. Compared to intra-sentential attention, however, the long-range dependency in document-level attention calculation inevitably introduces meaningless contextual noise, resulting in significant performance deterioration. To address this problem, this paper proposes a detection-based attention alignment method, to help each translating word focus on relevant informative contextual words. We first introduce a context detector that automatically evaluates each source-side word’s effect on the model’s prediction. Based on the detection results, we align the original attention weights by integrating the cosine similarity between the aligned and original attention weights into the loss function, under a multi-task framework, which allows DocNMT to more effectively capture the document-level context. The results for three English-German (En-De) public translation datasets show that the proposed method can obtain consistent improvements over a strong G-Transformer baseline.

Index Terms: document-level neural machine translation, contextual information, attention alignment

1. INTRODUCTION

During the last decade, neural machine translation (NMT) has made remarkable progress to become a state-of-the-art method, especially for sentence-level translation [1, 2]. In document-level translation, it is widely accepted that the introduction of discourse dependencies between sentences can improve the coherence and quality of the translated text [3, 4]. Like those for sentence-level NMT, most existing document-level NMT (DocNMT) models integrate contextual information using an attention mechanism. A general method for capturing this information is to encode a limited number of previous or neighboring sentences [5–8]. Intuitively, encoding the whole input document [9–11] as a single unit should provide the best integration of context information. As Bao et al. [9] pointed out, however, it is difficult to train such a DocNMT model. As inputs are sparse for long sequences in DocNMT, the attention weights from target to context in the source-side are flat, with large entropy values. This indicates that long context sequences confound attention on meaningful portions of current translating words, and the model is difficult to select valuable inputs from context sequences to navigate the redundancy of information.

The simplest way to tackle this is to shorten the contextual sequence in attention calculation. Some works proposed selecting several sentences in a document as context, through integrating sparse attention [12] or using an extra selection module [13, 14]. These methods, however, treat all the words in a sentence equally, without discrimination, and select the same sentences for them. This inevitably creates meaningless contextual noise. To address this problem, Xu et al. suggest that each word should focus on its own informative words, spread throughout the document [10].

Attention mechanism plays an important role in NMT by dynamically selecting relevant inputs for different predictions. Based on the assumption that the translation is improved if attention values are more accurate, Li et al. and Lu et al. proposed the refinement of attention distribution in sentence-level NMT models, through word alignment [15] and attention calibration [16] respectively.

Inspired by the works of Xu et al. [10] and Lu et al. [16], this paper proposes a novel detection-based attention alignment method, to refine target-to-context attention for DocNMT. First, we introduce a context detector (CD), which is a lightweight network attached to the backbone, and only exists in the training procedure. The CD automatically de-
etects the significance of each source-side word for model prediction. Based on the detection results, we then align the original document-level attention weights, and use the cosine similarity as an auxiliary function in model training. In the inference procedure, the CD is dropped, and only the conventional backbone is used. We carry out experiments on three commonly used DocNMT datasets for English-German (En-De) translation, covering the domains of TED talks, News, and Europarl from small to large. The results show that our proposed method can further improve DocNMT performance over a strong G-Transformer [9] baseline.

Main contributions of this paper can be summarized as follows:

- Our method can automatically evaluate the significance of each contextual word for model prediction by introducing a lightweight perturbation noise detector, which can be jointly optimized and removed when inference.
- We propose a attention alignment loss to help making each translating word more focus on its own relevant essential contexts, which is experimentally proved to reduce the entropy of contextual attention weights and outperform strong baseline.
- Our code is publicly available.

2. BACKGROUND

2.1. Document-level Neural Machine Translation

Formally, \( X = \{x_1, x_2, ..., x_S\} \) denotes the source document with \( S \) sentences, and \( Y = \{y_1, y_2, ..., y_T\} \) denotes the target document with \( T \) sentences. \( S \) and \( T \) are usually assumed equal, because sentences can be merged with sentence alignment algorithms [17] to fix mismatches. Compared with sentence-level NMT, DocNMT not only gets the benefit of intra-sentential information in parallel pair \( \{x_i, y_i\} \), but also takes advantages of the document’s contextual information. The translation probability from \( X \) to \( Y \) can therefore be represented as:

\[
P (Y | X) = \sum_{t=1}^{T} P(y_t | y_{<t}, X)
\]

(1)

2.2. G-Transformer

The G-Transformer is inherited from the Transformer [1]. After analyzing the training failure of the Transformer’s direct application into DocNMT, Bao et al. [9] attributed the problem to the huge complexity of target-to-source attention. They proposed the G-Transformer to reduce the hypothetical space of attention from target to source.

\[\text{GroupAttn}(\text{args}) = \text{softmax}(QK^T \sqrt{d_k} + M(\text{tag}))V\]

(2)

where function \( M(\cdot) \) works as an attention mask, excluding all tokens outside the sentence. This method allows local sentence representations to be obtained successfully.

On top layers, the model exploits standard multi-head attention as global attention (GlobalAttn), to capture contextual information of the document as a whole. GroupAttn and GlobalAttn are combined using a gate-sum module for integrating both intra-sentential and contextual information.

---

1 Code for the proposed attention alignment method is open-sourced at https://github.com/KangZhong99/attention-alignment
3. METHODOLOGY

To tackle the inaccuracy and flatness of attention distribution in DocNMT, we propose a detection-based attention alignment method to enhance attention weights, focused on the informative context in model training. As shown in Figure 2, we first design a context detector (CD) to deteriorate performance by modifying the original attention weights generated by the backbone network. Specifically: we obtain learnable flags, and then apply them to the original attention weights. The CD parameters are updated based on the loss function that maximizes performance deterioration. This allows us to locate source-side words that are informative or meaningless for translation. Finally, we design an attention alignment strategy to calibrate the original attention weights into the loss function under the multi-task framework.

3.1. Context Detector

The CD is a lightweight network attached to the G-Transformer backbone. The basic assumption of CD is that a decrease in the attention weights of informative words and an increase in those of meaningless words reduces translation performance. In this paper, we make a tentative modification on the original attention weights using learnable flags, which results in a deterioration of performance. Specifically, at the time step to produce t-th word in a sentence being translated, the learnable flag $f_t$ is generated, based on the hidden states $h_t^d$ from d-layer of DocNMT decoder ($d = 5$ or $6$ for G-Transformer):

$$f_t = \tanh(W^f \cdot h_t^d + b^f)$$

where $\tanh(\cdot)$ is the hyperbolic tangent function. $W^f$ and $b^f$ are trainable parameters vary among different attention layers and heads. The original attention weight $a_t$ can then be modified to $a_t^m$:

$$a_t^m = a_t \odot \bar{a}$$

where $\odot$ denotes an element-wise multiplication, and $\bar{a}$ denotes a uniform distribution (an average vector of attention heads). Qualitatively, a positive flag results in the increase of the attention weight, while a negative flag results in a decrease. The modification operation aims to make smallest variations of attention weights that lead to most obvious performance deterioration. According to this purpose, the objective function of CD is designed as:

$$\mathcal{L}(\theta^f) = \arg \min_{f_t \in [-1, 1]} \| f_t \|_2 - \mathcal{L}_{DocNMT}(a_t^m, \theta)$$

where $\theta^f = \{W^f, b^f\}$ is the parameters of CD and $\theta$ is the parameters of the original G-Transformer backbone. $\mathcal{L}_{DocNMT}(a_t^m, \theta)$ is the cross entropy loss of the DocNMT model based on the modified attention weight $a_t^m$. The hyper-parameter $\alpha$ serves as a regular term, encouraging most of the flags to be turned off, and alleviating the vanishing gradient problem of the hyperbolic tangent function. According to the second term, in order to deteriorate model prediction, the CD decreases the attention weights of informative words and increases those of meaningless words by setting corresponding flags to negative or positive, respectively.

3.2. Attention Alignment in Backbone Training

The CD can evaluate the importance of the source-side inputs for each output word. As analyzed above, a more negative (or positive) flag means the more beneficial (or harmful) the impact of corresponding contextual word. Here, we use the flag matrix $f_t$ as supervised information to align the original attention weight $a_t$ to aligned attention weight $a_t^a$.

$$a_t^a = a_t \odot e^{-f_t}$$

It is clear that the values of $a_t^a$ increase in informative contextual words and decrease in meaningless ones. This is consistent with our expectations for accurate attention. $a_t^a$ differs from $a_t^m$ in Eq.4. the former has a positive effect on the final translation, while the latter has a negative effect.

We then use cosine similarity between $a_t^a$ and $a_t$ to construct an auxiliary alignment loss function:

$$\mathcal{L}_a(\theta) = -CS(a_t^a, a_t)$$

Considering the attention mechanism is not well-trained at the early stage, and the final performance evaluation is contributed by the main training objective $\mathcal{L}_{DocNMT}(a_t, \theta)$ , the DocNMT model is optimized by:

$$\mathcal{L}(\theta) = \mathcal{L}_{DocNMT}(a_t, \theta) + \beta(s) \cdot \mathcal{L}_a(\theta)$$

$$\beta(s) = e^{-s/\tau}$$

where $s$ donates the training step, and $\tau$ is the decay time constant. A larger $\tau$ forces the model optimization to be affected by $\mathcal{L}_a$ for a longer time.

3.3. Training and Inference Issues

The proposed CD and backbone DocNMT model can be trained jointly. In a mini-batch, we first train the CD to find informative and meaningless contexts with the loss function in Eq.5, and then optimize the global attention mechanisms of backbone DocNMT following the objective function in Eq.8.
Table 1. Experiment results on three EN-DE datasets, where s-BLEU represents BLEU score for sentences, and d-BLEU [18] is the score for documents. Improvements over the sentence-level baseline are reported as "()".

<table>
<thead>
<tr>
<th>Model</th>
<th>TED</th>
<th>News</th>
<th>Europarl</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s-BLEU</td>
<td>d-BLEU</td>
<td>s-BLEU</td>
<td>d-BLEU</td>
</tr>
<tr>
<td>SAN [12]</td>
<td>24.42</td>
<td>-</td>
<td>24.84</td>
<td>-</td>
</tr>
<tr>
<td>Flat-Transformer [8]</td>
<td>24.87</td>
<td>-</td>
<td>23.55</td>
<td>-</td>
</tr>
<tr>
<td>Our Sentence Baseline</td>
<td>24.79</td>
<td>-</td>
<td>25.28</td>
<td>-</td>
</tr>
<tr>
<td>G-Transformer [9]</td>
<td>25.10 (+0.31)</td>
<td>27.17</td>
<td>25.58 (+0.30)</td>
<td>27.11</td>
</tr>
<tr>
<td>Attention Calibration</td>
<td>24.97 (+0.18)</td>
<td>27.12</td>
<td>25.20 (-0.08)</td>
<td>26.89</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>25.47 (+0.68)</td>
<td>27.65</td>
<td>26.09 (+0.81)</td>
<td>27.66</td>
</tr>
</tbody>
</table>

In practice, the parameter increments of CD are negligible (2M per layer and 4M in total), and can be removed in the inference procedure.

4. EXPERIMENTS

4.1. Datasets and Settings

We conduct experiments on widely used document-level parallel benchmark datasets, including three domains for English-German (En-De) translation: TED [19], News [20] and Europarl [21]. We split the documents into instances with up to 512 tokens. Moses [22] is used for data processing, and BPE [23] is used with vocab-size of 30K merges. Detailed statics for these datasets are shown in Table 2.

<table>
<thead>
<tr>
<th>Data</th>
<th># of Docs</th>
<th># of Sents/Doc</th>
<th># of sents/Doc in our Inst</th>
</tr>
</thead>
<tbody>
<tr>
<td>TED</td>
<td>1.7K/92/22</td>
<td>123/98/105</td>
<td>18.3/18.5/18.3</td>
</tr>
<tr>
<td>News</td>
<td>6K/80/154</td>
<td>40/25/20</td>
<td>12.8/12.6/11.3</td>
</tr>
<tr>
<td>Europarl</td>
<td>118K/239/359</td>
<td>14/15/14</td>
<td>10.3/10.4/10.3</td>
</tr>
</tbody>
</table>

The DocNMT model is trained in two stages: first, a vanilla Transformer [1] base model is trained for sentence-level translation, then the DocNMT model is finetuned based on sentence baseline with document-level data. We use the same model configuration as Bao et al. [9] and train all models on 2 GeForce RTX 3090 GPUs. To finetune on the sentence baseline, we employ Adam ($\beta_1 = 0.9, \beta_2 = 0.998$) for parameters optimization with warm-up steps of 2,000 for TED and 4,000 for News and Europarl. The dropout rate is 0.3 except for News (0.4). The models are trained with the batch size of 32K tokens for all datasets. The batch size of 32K tokens is achieved by using the batch size of 4096 tokens and updating the model for every 8 batches. Following Bao et al. [9], we apply word dropout [24, 25] to the inputs with $p = 0.1$. We set hyper-parameters as $\alpha = 1.5$ and $\tau = 10^4$ in Eq.5 and Eq.8 respectively.

We mainly build four systems for comparison: the Sentence Baseline, the G-Transformer [9], Attention Calibration [16] and the Proposed Method.

4.2. Results and Analysis

The results of our experiments are listed in Table 1, along with those of other DocNMT models. Finetuned on a strong sentence-level baseline, the performance of G-Transformer improves on three datasets by 0.54 s-BLEU points on average. Applying the proposed method to G-Transformer to refine contextual attention calculation further improves the gain from 0.54 to 0.95 s-BLEU points, and achieves the best results among all those listed on the two lager datasets, News and Europarl.

The hyper-parameter $\alpha$ in Eq.5 decides the values of generated flags, and influences the concentration degree of global attention by Eq.8. The entropy values of global attention and s-BLEU on News, with respect to different $\alpha$ values, are depicted in Figure 3. The s-BLEU score presents a negative association with contextual attention entropy, and reaches its climax when $\alpha$ is 1.5. This verifies our hypothesis that translating the current word only replies on very few contextual words and that sharpening the focus of document-level attention is beneficial to model prediction.

Figure 4 (a) depicts the entropy curves, with respect to different training epochs on vanilla G-Transformer and the proposed method. Both curves show a convergence trend in global attention. The curve of the former retains stable during training, however, which means the model captures a significant amount of meaningless contextual information. As for the proposed method, the curve falls sharply for about 20 epochs, and then retains at a relatively low level. Figure 4 (b) plots the attention distribution of final converged models. In our method, the model is more confident about the choice of informative contextual words, and focuses on very few of them compared to the vanilla G-Transformer.
To further investigate whether our model is able to improve the translation of discourse phenomena, we conducted linguistic evaluation on deixis and ellipsis using an English-Russian (En-Ru) linguistic contrastive test set [26]. To make a fair comparison, we follow [26] to use 6M sentence-level instances to train sentence-level baseline and 1.5M document-level instances to train our models (G-Transformer and the proposed method). Results are reported in Table 3. The proposed method achieves improvements on deixis and ellipsis (VP) compared with G-Transformer. This indicates that our method can make better use of context to deal with discourse phenomena.

The upper box of Table 4 shows an example where our method correctly translates the pronoun “it” (highlighted in bold). To analyze the effect of our attention alignment approach, we print two particular attention heads, one from the G-Transformer and one from our method. The source-side surrounding contexts with assigned attention weights at the time step after the prediction “teilt” are shown in the lower box of Table 4. Both heads focus more on the contextual reference “love”, which is related to the query “it”. In contrast, our method reduces the disturbance of irrelevant context words, like “Remi”, and “differences”.

### Table 3. Linguistic evaluation results on the contrastive test set (accuracy).

<table>
<thead>
<tr>
<th>Model</th>
<th>deixis</th>
<th>ellipsis (infl.)</th>
<th>ellipsis (VP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sent†</td>
<td>50.0</td>
<td>53.0</td>
<td>28.4</td>
</tr>
<tr>
<td>concat†</td>
<td>83.5</td>
<td>47.5</td>
<td>76.2</td>
</tr>
<tr>
<td>CADec†</td>
<td>81.6</td>
<td>58.1</td>
<td>76.2</td>
</tr>
<tr>
<td>G-Transformer</td>
<td>89.7</td>
<td>84.7</td>
<td>82.2</td>
</tr>
<tr>
<td>proposed method</td>
<td><strong>90.3</strong></td>
<td><strong>84.9</strong></td>
<td><strong>83.5</strong></td>
</tr>
</tbody>
</table>

† indicates that the results are borrowed from the original paper [26].

### Table 4. Example of pronoun disambiguation. In German, if the reference of “it” is feminine (e.g., “love”), “sie” is used. Otherwise, it is “es” (masculine) or “es” (neutral). The intensity of color represents the attention given to a specific word.

<table>
<thead>
<tr>
<th>Attention to reference word “love”</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-Transformer (Head 3):</td>
</tr>
<tr>
<td>... Remi knows what love is. He shares it regardless.</td>
</tr>
<tr>
<td>He doesn’t care about religious differences, and gets ...</td>
</tr>
<tr>
<td>Proposed Method (Head 3):</td>
</tr>
<tr>
<td>... Remi knows what love is. He shares it regardless.</td>
</tr>
<tr>
<td>He doesn’t care about religious differences, and gets ...</td>
</tr>
</tbody>
</table>

5. CONCLUSION

In this paper, we propose a detection-based attention alignment method to force the DocNMT model to increase the focus of each word on relevant key contextual words. We design a lightweight context detector to evaluate the importance of each contextual word, which proves simple yet effective. Then, we introduce a strategy that uses this supervised information to guide attention alignment. As demonstrated, the proposed method observably reduces the entropy values of document-level attention, and consistently improves on the strong G-Transformer baseline. More encouragingly, our work provides valuable reference on self-supervised learning for improved or more-focused attention in other long-input generation frameworks.

### References


Abstract—To improve the efficiency of fixing bugs, mobile application developers must reproduce bugs reported by testers or users as quickly as possible. Automated testing tools can help but are not designed for reproducing bug reports. To improve the efficiency of reproducing crashes, we propose a widget hierarchy graph guided crash reproduction method for Android apps. It builds a widget hierarchy graph, locates suspicious widgets using bug reports and project files, calculates widget fitness, and guides automated testing to reproduce crashes quickly. To evaluate the effectiveness of the proposed method, experiments are conducted on real Android application bug reports and compared with the automated testing tools APE and PUMA. Experimental results show that our method successfully reproduces six bug reports that cause Android app crashes. In addition, compared with APE and PUMA, the average time for our method to reproduce crashes decreased by 51.94% and 71.47%.

Index Terms—reproduce crashes, bug reports, Android applications, widget hierarchy graph

I. INTRODUCTION

As of Dec 2022, Android accounted for 72.37% of the mobile operating system market [1]. With its growing popularity, Android applications have proliferated. Developers face fierce competition and must release new versions of applications quickly, making thorough testing difficult. This results in bugs in released versions, increasing testing and maintenance costs and challenging app robustness and reliability.

Bugs in mobile applications result in user loss [2], so developers must quickly respond and fix them, especially application crashes that directly affect usability [3]. Reproducing crashes may require complex operations and can be inefficient when done manually. Automated testing tools can trigger some crashes but lack pertinence and efficiency. Software projects use bug tracking systems (such as Bugzilla1, GitHub Issue Tracker2, etc.) to manage testing and accelerate bug fixing. When reporting bugs, users and testers usually provide application versions, system versions, bug screenshots, stack traces, widget information, etc. Reviewing bug reports is an important way to find and reproduce crashes. By using information from bug reports, a developer can refine the search scope and allocates more testing resources to the locations related to the bug reports to improve crash reproduction efficiency. However, existing automated testing tools rarely pay attention to bug reports.

*Zhanqi Cui is the corresponding author.
1Bugzilla. https://bugzilla.mozilla.org/describekweds.cgi
2GitHub Issue Tracker. https://github.com

This paper makes following contributions.
- proposes a widget hierarchy graph guided crash reproduction method for Android applications, to improve the efficiency of reproducing crashes by using widget hierarchy graphs;
- experiments are carried on a set of real applications to evaluate the effectiveness of the proposed method.

II. WIDGET HIERARCHY GRAPH GUIDED CRASH REPRODUCTION METHOD FOR ANDROID APPLICATIONS

Fig. 1 shows the framework of the widget hierarchy graph guided crash reproduction method for Android applications. It consists of three components: (1) Build Widget Hierarchy Graph, (2) Calculate Fitness of Widgets, (3) Generate Test Script. These components are discussed in the following sections.

It is possible to increase the efficiency of reproducing a crash by using the bug report information effectively and concentrating more testing resources on suspicious widgets. Therefore, we propose a widget hierarchy graph guided crash reproduction method for Android applications. By automatically analyzing the project file of the application under test, it creates a widget hierarchy graph, which is used in combination with the bug report to generate test scripts for reproducing crashes. To evaluate the effectiveness of our method, we implement the prototype tool based on this method and conduct experiments on 6 bug reports of 5 Android applications. We compare it with advanced automated testing tools APE [4] and PUMA [5]. The experimental results show that our method and APE both reproduced the application crashes mentioned in the 6 bug reports, while PUMA only reproduced 5. Compared with PUMA and APE, our method saves 71.47%, 51.94% time to reproduce the crashes in average, respectively.

Fig. 1: The Framework of Widget Hierarchy Graph Guided Crash Reproduction Method for Android Applications.
A. Build Widget Hierarchy Graph

We use a widget hierarchy graph to describe the relationships between widgets in an Android application. A widget hierarchy graph can be defined as follows.

Definition 1. (Widget Hierarchy Graph). A widget hierarchy graph of Android application $A$ is a 5-tuple $G = (V, C, J, W, P)$, where:

- $V = \{v_1, v_2, \ldots, v_i, \ldots\}$ is the set of nodes in the graph, and the node $v_i \in V$ is a function in $A$;
- $C = \{c_{1,2}, c_{3,4}, \ldots, c_{k,l}, \ldots\}$ and $J = \{j_{1,2}, j_{3,4}, \ldots, j_{m,n}, \ldots\}$ are the sets of two kinds of edges in the graph, the edge $c_{k,l} \in C$ is a function call in $A$, and the edge $j_{m,n} \in J$ is an interface jump in $A$;
- $W$ is the set of widgets in $A$;
- $P$ is the set of interfaces in $A$, and each interface consists of several widgets.

For the function $v_i \in V$, $W_i = \text{Annotation}(v_i) \subseteq W$ is the set of widgets related to the function $v_i$, indicating that the function $v_i$ will be called when operating any widget in $W_i$, if $\text{Annotation}(v_i) = \emptyset$, means that the function $v_i$ has no related widgets. For the function call $c_{k,l} \in C$, $\text{Caller}(c_{k,l}) = v_k \in V$, $\text{Callee}(c_{k,l}) = v_l \in V$, indicating that the function $v_l$ in $A$ can call the function $v_k$. For the interface jump $j_{m,n} \in J$, $\text{From}(j_{m,n}) = p_m \in P$, $\text{To}(j_{m,n}) = p_n \in P$, indicating that the interface $p_m$ in $A$ can jump to the interface $p_n$, where the starting point of the jump is $p_m$ and the ending point of the jump is $p_n$. $\text{Trigger}(j_{m,n}) = w^m_n \in p_m$, indicating that the widget $w^m_n$ in the interface $p_m$ is the widget that triggers the interface jump to $p_n$.

To build the widget hierarchy graph, we analyze the APK file of the application under test for function calls. For the Android applications $A$, the set of functions $V$ and the set of function calls $C$ can be obtained directly from the function call. FlowDroid\(^3\) is used to get function calls. To simplify the analysis of Android applications, FlowDroid generates dummy methods that represent the order of implicit invocations of lifecycle callback methods and GUI callback methods in applications.

After getting the set of functions $V$ and the set of function calls $C$, we analyze the project file of the application under test in combination with function calls to obtain interface jumps and the relationships between widgets and functions. First, get the set of widgets $W$ directly from the project file of the application under test. Then, analyze the project file to obtain the method related to the set of widgets $W$ (when operating one widget, if the widget automatically calls one method, it is said that the widget is related to the method). For the function $v_i \in V$, $W_i = \text{Annotation}(v_i) \subseteq W$, that is, the function $v_i$ and the widgets in $W_i$ are related to each other. Next, traverse the set of functions $V$, if the function $v_u$ is a dummy function, obtain the set of widgets $W_u = \text{Annotation}(v_u) \subseteq W$ related to the functions called by $v_u$, and add the interface $p_u$ where $W_u$ is located to the interface set $P$. Finally, if the interface $p_m$ can jump to the interface $p_n$, add the interface jump $j_{m,n}$ to $J$, where $\text{From}(j_{m,n}) = p_m$, $\text{To}(j_{m,n}) = p_n$.

B. Calculate Fitness of Widgets

Our method guides the automated testing by using the fitness of each widget. The fitness of widgets can be obtained through widget hierarchy graph and suspicious widgets. To get the set of suspicious widgets $W_{\text{susp}}$, it uses the information of widgets or exception trace stack contained in the bug report to find the widgets related to the bug report from the project file of the application under test.

Algorithm 1 Calculate Fitness of Widgets for Application

Input: Set of suspicious widgets $W_{\text{susp}}$, widget hierarchy graph $G = (V, C, J, W, P)$

Output: Fitness $F$

1. $F \leftarrow \emptyset$
2. $P_{\text{locate}} \leftarrow \text{FindPageOfSuspWidget}(W_{\text{susp}}, G)$
3. for each page $p_m \in P_{\text{locate}}$ do
   4.    for each widget $w^m_n \in p_m$ do
   5.       $F.add(w^m_n, \text{CalFitWithFormula1}(w^m_n, W_{\text{susp}}))$
   6. while first cycle or $p_{\text{start}} \neq \emptyset$ do
   7.       $p_{\text{start}} \leftarrow \text{FindStartingPageOfJump}(P_{\text{locate}}, G)$
   8.       if $p_{\text{start}} \neq \emptyset$ then
   9.          for each page $p_n \in P_{\text{start}}$ do
   10.             for each widget $w^m_n \in p_n$ do
   11.                $F.add(w^m_n, \text{CalFitWithFormula2}(w^m_n, G))$
   12.       $P_{\text{locate}} = P_{\text{start}}$
13. return $F$

Algorithm 1 describes the process of calculating the fitness of widgets. The input is the set of suspicious widgets $W_{\text{susp}}$ and the widget hierarchy graph $G = (V, C, J, W, P)$, the output $F$ is the fitness values of widgets. After initializing the set $F$ (line 1), we traverse all the interfaces in the widget hierarchy graph, find the interfaces to which all suspicious widgets belong, and record the set of interfaces $P_{\text{locate}}$ that contains suspicious widgets (line 2). Lines 3 to 5 traverse the widgets in each interface in $P_{\text{locate}}$. Calculate the fitness of these widgets, and save the results in $F$. Next, we continuously searches the set of starting points of interface jumps and calculates the fitness of widgets in the starting points (lines 6-12), until the set is empty. In each cycle, we first traverses all interface jumps in the widget hierarchy graph, finds the interfaces which take any interface in the set of interfaces built in the previous iteration as the ending points of interface jumps, and save the found interfaces as the set $p_{\text{start}}$ (line 7). If $p_{\text{start}}$ is not empty, traverse the widgets in each interface of $p_{\text{start}}$, then take the interfaces in $p_{\text{start}}$ as ending points of the jumps to proceed next iteration after calculating the fitness of the widgets (lines 9-11); if $p_{\text{start}}$ is empty, indicating that there is no interface that is the starting point of an interface belonged to the $p_{\text{start}}$ set, the algorithm ends the iteration and exports $F$ which saves the fitness values of widgets (line 13).

For the interface $p_n \in P_{\text{locate}}$ which the suspicious widget belongs to (line 2), we uses the formula (1) to calculate the fitness of each widget in $p_n$ (line 5). To make the suspicious widget in $p_n$ more likely to be covered during the testing, for the widget $w^m_n \in p_n$, if $w^m_n \in W_{\text{susp}}$, then its fitness is $K \times N$; if $w^m_n \notin W_{\text{susp}}$, then its fitness is $N$. Among them, $N$ is a non-zero constant, and $K$ is a constant greater than 1.

\(^3\)FlowDroid. https://github.com/secure-software-engineering/FlowDroid
For the interface $p_m \in P_{\text{start}}$ (line 7), there are $p_n \in P_{\text{locate}}$ and $j_{m,n} \in J$, that $\text{From}(j_{m,n}) = p_m$ and $\text{To}(j_{m,n}) = p_n$. We uses the formula (2) to calculate the fitness of widgets in $p_m$. Widgets that can trigger interface jumps may lead the testing to the interface that triggers the application crash. In order to make such widgets more likely to be covered during testing, for the $w^m_u \in p_m$, if $w^m_u$ is the widget in $p_m$ that can trigger the interface jump to $p_n$, its fitness is the sum of the fitness of all widgets in $p_n$, that is $\sum_{w^m_n \in p_n} \text{Fit}(w^m_n)$, if $w^m_u$ in $p_m$ is not the widget that can trigger the interface jump, its fitness is $N$.

C. Generate Test Script

When the calculation of fitness for widgets is completed, automated testing can be guided with the fitness to reproduce the crashes more quickly. During testing, our method continuously selects and operates widgets in current interface of the application based on the fitness of widgets within a specified time limit. Specifically, we use formula (3) to calculate the fitness of any widget $w^t_y$ within the current interface. In formula (3), $\text{Fit}(w^t_y)$ is the fitness of the widget $w^t_y$ in current interface $p_t$, and $\sum_{w^t_z \in p_t} \text{Fit}(w^t_z)$ is the sum of the fitness of all widgets in $p_t$. The widget $w^t_u \in p_t$, the probability of being selected to be operated is the ratio of its fitness to the sum of the fitness of all widgets in $p_t$. After selecting and operating a widget in current interface according to the probability, the operation is recorded in the test script. Our method ends the testing when the maximum test time is reached or the application crash is successfully reproduced, and outputs the test script. To determine whether the application crash has been successfully reproduced, the method compares the log with the stack trace in the bug report.

$$P_b(w^t_y) = \frac{\text{Fit}(w^t_y)}{\sum_{w^t_z \in p_t} \text{Fit}(w^t_z)} \quad (3)$$

III. EXPERIMENTS AND EVALUATIONS

A. Experimental Design

Our method aims to use the information in the bug report to improve the efficiency of crash reproduction, so how effective and efficient is CrPDroid compared to other automated testing tools? We compare our method with PUMA and APE for crash reproduction. APE and PUMA are both model-based Android application testing tools. We measures effectiveness by the number and time costs to reproduce the crash described in the bug report within a limited time. The other settings of the experiment are as follows:

1) Experimental Settings: In the experiments, we follow the experimental settings of Zhao et al. [6] and limit the testing time of each experiment to 2 hours. To get more accurate time of reproducing crashes, each testing tool is set to run 10 times on each experimental subject. The average of the results will be recorded as the final time. In addition, after conducting small-scale experiments on our method, we set its parameter $K$ to 45.

2) Experimental Subjects: The experimental subjects are obtained from Q-testing [7]. We analyze the above 50 applications in Q-testing. First, search for “Crash” on the bug-tracking systems of the 50 applications, and collect 27 bug reports. Then, manually verify and reproduce the crashes described in the bug reports, and remove the bug reports due to failure to build APK, environmental problems and unreproducible. As shown in TABLE I, six bug reports are obtained from five applications.

3) Implementation and Environment: Based on the proposed method, a prototype tool is implemented on the basis of PUMA framework to evaluate the effectiveness. The development and executing environment of the tool is 16GB memory, 6-core 3.3GHz CPU, Ubuntu20.04, Android SDK (4.3-8.0), JDK1.8.0.

B. Experimental Results

Fig. 2 shows the result of reproducing crashes by our method and automated testing tools PUMA, APE. Our method and APE can reproduce all crashes, but APE takes longer than our method. PUMA can reproduce five crashes, except for the crash in BetterBatteryStats. For these five crashes, our method and PUMA take same time to reproduce the crash in RadioBeacon (Bug1), while, our method takes less time to reproduce the other four crashes. Our method saves 71.47% time to reproduce crashes than PUMA in average. The reason is that the model-based exploration strategy of PUMA, which doesn’t consider suspicious widgets during exploration, is less efficient in reproducing crashes. With further analysis of the crash in BetterBatteryStats which cannot be reproduces by PUMA, we found that PUMA triggered an exception during backtracking the application, which cause PUMA terminated and fail to reproduce the crash.

As shown in the experimental result, our method successfully reproduces the crashes in the bug reports, and outperforms PUMA and APE in terms of time costs.

C. Validity Analysis

1) External Validity: External validity threats arise from the representativeness of the selected evaluation subjects and bug reports on the one hand and the generality of our method on the other. To ensure the representativeness of the evaluation subjects and bug reports, the Android applications selected in the experiment are widely used in related testing works [7] [8], and the source codes are all open source on GitHub or F-Droid.
Bug reports are obtained from the respective bug tracking system or comment section of the application. To improve the generality of our method, we implement the method based on PUMA. PUMA has been validated on 3600 apps in Google Play [5] and is used as the base framework in the related works of Liu et al. [9].

2) Internal Validity: The internal validity threat mainly comes from the accuracy of the constructed widget hierarchy graph and the correctness of the exploration of the application under test. To improve the accuracy of the widget hierarchy graph, we use the static analysis tool FlowDroid to analyze and obtain function calls from the APK file. FlowDroid is widely used for data flow analysis of Android applications and Java programs. PUMA framework is used to ensure the exploration of applications under test is accurate for our method. PUMA has been widely used in analyzing program attributes (such as application state, widget information, etc.) [5]. Moreover, we check and test the implementation code for constructing widget hierarchy graphs and calculating fitness of widgets to minimize the risk of validity.

IV. RELATED WORK

Currently, many works focused on GUI testing and the importance of bug reports in the quality assurance of Android applications.

Monkey [10] is the most commonly used random strategy based automated testing tool, which generates a pseudo-random stream of GUI events by randomly interacting with screen coordinates. The random strategy used by Monkey performs well on some benchmark applications. FUSION [11] assists users in automatically generating operation steps for reproducing bug reports by dynamically analyzing GUI events of Android applications. With FUSION, users can create more comprehensive and accurate bug reports, and developers can get operable information from bug reports, which could help reproduce and fix Android application bugs.

V. CONCLUSION

In this paper, we present a widget hierarchy graph guided crash reproduction method for Android apps. It creates a widget hierarchy graph by analyzing the project file and uses it with bug reports to generate test scripts for reproducing crashes. Experimental results show our method outperforms PUMA and APE in time costs. In the future, we plan to use information retrieval-based bug localization to improve the efficiency of reproducing crashes.

ACKNOWLEDGEMENT

This work was supported in part by the Jiangsu Provincial Frontier Leading Technology Fundamental Research Project (BK20202001), the National Natural Science Foundation of China (No. 61702041), and the Beijing Information Science and Technology University “Qin-Xin Talent” Cultivation Project (No. QXTCP C201906).

REFERENCES

Construction Site Fence Recognition Method Based on Multi-Scale Attention Fusion ENet Segmentation Network

Xing Zhang  
Hubei Key Laboratory of Intelligent Vision Based Monitoring for Hydroelectric Engineering  
China Three Gorges University  
Yichang, China  
zhangxing@ctgu.edu.cn

Tinglong Tang*  
Hubei Key Laboratory of Intelligent Vision Based Monitoring for Hydroelectric Engineering  
China Three Gorges University  
Yichang, China  
tangtinglong@ctgu.edu.cn

Yirong Wu  
Hubei Key Laboratory of Intelligent Vision Based Monitoring for Hydroelectric Engineering  
China Three Gorges University  
Yichang, China  
yirongwu@hotmail.com

Tingwei Quan  
Wuhan National Laboratory for Optoelectronics  
Huazhong University of Science and Technology  
Wuhan, China  
quantingwei@hust.edu.cn

Abstract—In this paper, we propose a fence recognition method based on the ENet (Efficient neural Network) segmentation network to address the problems of traditional segmentation networks, which have poor performance in recognizing fences with a large range of scale variations and hollow structures. Firstly, a multi-scale attention fusion ENet segmentation network is designed, which is trained using the fence with obvious color features. Then, a morphological algorithm is used to process the predicted image to restore the fence segmentation results. The designed multi-scale attention fusion segmentation network performs better on fence datasets than traditional methods. In addition, the activation function Leaky_Relu6 further enhances the stability and generalization ability of the network. The experiments are conducted on 540 fence images from different construction sites, and the computed IoU is 90%. The processing speed is about 28 frames per second. The experimental results show that our proposed network outperforms traditional segmentation algorithms in fence recognition performance, and achieves robustness in different construction scenarios while meeting the requirements of both accuracy and speed.

Keywords—fence recognition; ENet; multi-scale attention; morphological algorithms;

I. INTRODUCTION

As a common safety protection facility on the construction site, the fence is usually used to isolate dangerous areas from the construction staff, which can play a certain protective role. However, when the staff is facing away from the fence, there will still be some safety hazards due to the blind spot. Therefore, this paper hopes to apply the method of deep learning to identify the fence on the site in real-time and accurately, and send a reminder when the construction staff and the fence are close, which can reduce the accident rate to a certain extent and ensure the safety of construction staff.

The construction site fence is mainly composed of thin red and white bars arranged in a certain shape, and its main shapes include vertical and cross-shaped patterns, as shown in Fig. 1. Traditional methods of acquiring segmented datasets involve pixel-by-pixel annotation of foreground pixels. However, pixel-by-pixel operations would require a lot of time due to the high density of the fence, and the hollow structure of the fence itself would contain a large amount of background information, making it difficult for the network to learn the correct features. Moreover, because the relative scales of the fence in different images can vary dramatically, the range of scale changes in the feature map is also large, which presents a challenge for traditional segmentation algorithms.

As technology continues to evolve and replace older methods, there have been significant improvements in the accuracy and speed of multiscale object segmentation. However, not all segmentation algorithms can be applied to practical engineering projects. While accuracy and speed are important, other factors such as memory usage and model stability must also be considered. ENet (Efficient neural Network) [1] is a lightweight segmentation network that has a higher inference speed than traditional networks, making it more suitable for environments with limited memory, such as embedded devices.

Considering the structural characteristics of fence data, this paper proposes a fence recognition method based on improved ENet neural network. Firstly, a multi-scale attention fusion ENet neural network is constructed. Then, a new activation function is designed for the encoding part of the network to increase the model's stability and generalization ability. Finally, the network is trained using parts of the fence structure with clear color features, and the predicted image is processed using morphological algorithms to restore the entire fence structure. The proposed network has the following advantages:

DOI Reference Number: 10.18293/SEKE23-081
II. RELATED WORK

A. Multi-scale Segmentation

In recent years, image segmentation has been widely used to identify target objects in images or videos and perform instance segmentation at the pixel level [4]. However, the segmentation of images with large-scale variations remains a challenging problem [5]. Olaf et al. proposed the U-Net [6] segmentation network, which uses an encoding-decoding network structure and skips connections in the encoding and decoding parts to achieve good performance in multi-scale medical image segmentation. Chen et al. developed the Deeplab [7] segmentation network, which uses dilated convolution and fully connected convolutional neural networks to further improve segmentation accuracy. Long et al. proposed the FCN [8] segmentation algorithm based on fully convolutional neural networks, which uses deconvolutional layers to output more refined image segmentation results and employs skip connections to increase the network's robustness. Zhang et al. employed a context encoding [9] network to obtain global information of the target and combined it with a common encoding-decoding network structure to output feature maps with richer multi-scale information.

B. Scene Segmentation

Yu et al. developed the BiSeNet [10] lightweight semantic segmentation network, which uses a bidirectional mechanism to acquire receptive fields that retain spatial and semantic information, allowing for faster network speeds without sacrificing accuracy. Mohan et al. proposed the EfficientPS [11] panoramic segmentation network, which can simultaneously segment background and foreground information. They introduced a new panoramic fusion module that dynamically adjusts the panoramic segmentation results based on the confidence levels of semantic and instance segmentation, greatly aiding autonomous driving environment perception. Liu et al. proposed the CRNet [12] segmentation network with a cross-reference mechanism that enhances model output feature representations by comparing similar features in two images, achieving small-sample image segmentation. Weng et al. introduced the DMA-Net [13] semantic segmentation network suitable for street view data, which aggregates feature maps generated by different convolutional layers through a multi-branch aggregation network to obtain multi-scale information of the target, achieving good performance on the CamVid dataset.

C. Segmentation with Attention Mechanism

Fu et al. proposed a dual attention mechanism [14] that combines channel and spatial attention to the aggregate context information and improves the expression ability of multi-scale feature maps, resulting in improved performance in scene segmentation. Hou et al. proposed the self-attention distillation network SAD [15], which optimizes low-level learning with high-level positional information and enhances high-level feature expression with low-level learned attention features, performing well in lane line segmentation. Tao et al. utilized a hierarchical multiscale attention mechanism [16] to predict different scale targets and output final results through pixel level operations, reducing memory usage by four times while improving accuracy and segmentation speed. Huang et al. developed CCNet [17] which uses a criss-cross attention module that cycles through the network to obtain horizontal and vertical contextual information of target pixels and generate richer feature maps.

III. METHOD

In this paper, we propose a fence recognition method based on an improved ENet neural network. Firstly, we construct a multi-scale attention fusion ENet neural network, and design a new activation function Leaky_Relu6 in the encoding part to replace PReLU [18], which improves the stability and generalization ability of the network. Then, we train the network using fences with obvious color features and restore the complete fence area from partially recognized fence structures through computer morphological algorithms.

A. ENet with Multi-Scale Attention Fusion

To effectively reduce the computational cost and time in segmentation tasks, this paper chooses the ENet segmentation network as the main component of the segmentation network. The ENet network structure is mainly composed of an initial module and several bottleneck modules, as shown in Fig. 2.
In the initial block in Fig. 2, a 3x3 convolution with a stride of 2 and a maximum pooling are performed on the input. The results on both sides are merged into channels by concatenation. In the main branch of the bottleneck module in Fig. 2, the channel number is reduced by a 1x1 convolution, and a specific number of feature maps is output by a 1x1 convolution layer. If the bottleneck module is not down-sampled in the other branch, no operation is performed in this branch and it is directly added to the main branch. If it is down-sampled, this branch is first down-sampled by maximum pooling, then padded to achieve the same size as the feature map in the main branch.

The original ENet neural network consists of five parts, its structure is similar to an encoding-decoding structure, in which the first three parts extract feature information, and the last two parts are used to restore the feature map with the original size.

Generally speaking, for neural networks, the lower the feature map size and the less down-sampling operations, the richer the location information of small targets. The higher the feature map size and the larger the receptive field, the richer the semantic information. However, as the number of down-sampling operations increases, the location information of small targets becomes insufficient. Therefore, this paper combines the feature maps of the first and fourth parts of ENet, as well as the feature maps of the second and third parts, through a multi-scale fusion to combine the location information of the lower layers with the semantic information of the higher layers, thereby enhancing the model's feature learning ability.

At the same time, in each fusion process, in order to better aggregate foreground pixels and reduce the influence of interference pixels, Coordinate Attention (CA) [19] is added during the fusion process. CA is an attention mechanism that embeds positional information into channel attention by dividing channel attention into two dimensions of encoding. One dimension is used to obtain the dependency relationship between foreground pixels and contextual information, and the other dimension is used to preserve the positional information of foreground pixels, thereby enabling the feature map to have better direction and position expression ability.

B. Leaky_Relu6 Activation Function

In the encoding part of the original ENet network, PRelu was used as the activation function instead of Relu. Compared to the case where the negative output of Relu is constantly 0, PRelu can adjust the output value of the negative part adaptively by a learnable parameter. However, this occurs at the cost of the need of learning an extra parameter. Therefore, this paper leans towards the use of Leaky_Relu for the negative output, as shown in Fig. 3a. It outputs a small value to retain some useful information and avoid dead neurons without the need for additional parameter learning.

For the positive output, considering the high requirements for devices and networks on mobile devices, allowing unlimited output may lead to gradient explosion. Therefore, inspired by Relu6 activation function, as shown in Fig. 3b, a maximum value of 6 is set as the upper limit to suppress unlimited output. This can effectively solve the gradient explosion problem, prevent overfitting, and improve the model's generalization ability and stability.

Inspired by Leaky_Relu and Relu6 activation functions, this paper proposes a new activation function called Leaky_Relu6, as shown in Fig. 3c. It outputs an extremely small value when the input is less than 0. When the input is between 0 and 6, it outputs the same value as the input, and when the input is greater than 6, it outputs 6. This activation function is also used in the encoding component of the network, to replace the PRelu activation function in the original ENet network.

Combined with the above improvement points, the structure diagram of the network we proposed in this paper is shown in Fig. 4.
The image containing fence facilities is fed into the trained network to obtain the predicted image, which is then subjected to morphological operations such as closing, polygon dilation, and rectangle filling [20] to recover the complete fence structure from partial fence segmentation results. At the same time, the position corresponding to the fence area in the construction site picture is marked to realize the positioning of the fence area of the construction site.

Firstly, a rectangular kernel of size 9*9 is set to divide the predicted image into several rectangular structural elements, which are then subjected to closing operation to fill the concave corners. The mathematical formula for the closing operation is as follows:

\[
A \cdot B = (A \oplus B) \odot B
\]  

where A represents the predicted image, B represents the rectangular structural elements segmented by the rectangular kernel, and \(\oplus\) represents the dilation operation. \(\odot\) represents the corrosion operation. The formula for dilation operation is as follows:

\[
P = (A \oplus B) = \{x, y | B_{x,y} \cap A \neq \emptyset\}
\]

where \(B_{x,y}\) means that the origin of the rectangular structural element is moved to the point \((x, y)\). The prediction image A is traversed by the structural element B, and if there is pixel intersection between B and A, the entire structural element B is retained. The corrosion operation is performed on the image P after the dilation operation, and the mathematical formula of the corrosion operation is as follows:

\[
A \cdot B = P \cap B = \{x, y | B_{x,y} \in P\}
\]

which means that structural element B goes through the binary image P after dilation operation. The pixels that intersect between structural element B and binary image P are retained, and other disjoint pixels are removed.

To prevent erroneous connections caused by closure operation, we set a vertical pixel threshold of 10. Columns with fewer than 10 pixels in the vertical direction are considered erroneous connections and are removed to prevent errors in subsequent steps. After polygon dilation, a rectangle filling operation is performed on the foreground pixels. Specifically, the number of pixels between the top and bottom pixels in each column are set to 255. Through these morphological image operations, the complete fence structure is successfully restored from the partially segmented fence structure. The implementation process of graphic processing algorithms is shown in Fig. 5.
segmentation that calculates the ratio of the intersection to the union of two sets. The formulas for calculating the IoU and v are as follows:

\[ \text{IoU} = \frac{\text{target} \cap \text{prediction}}{\text{target} \cup \text{prediction}} \]  

\[ v = \frac{N}{\sum t_i} \]  

In the above formulas, target represents true pixel area of the safety fence and prediction represents the pixel area that is actually predicted to be the security fence. N represents the total number of images, and t represents the time required to process each image.

C. Analysis and Comparison of Results

Firstly, based on the distribution characteristics of fence structures in construction site images, fences are classified into four types: vertical fences without occlusions, intersecting fences without occlusions, vertical fences with occlusions, and intersecting fences with occlusions. These four types are respectively denoted as type A, B, C, and D.

To test the effectiveness of the method proposed in this paper, the segmentation of fences of the above four types was tested individually (Fig. 6). Specifically, Fig. 6a displays the original images of fences captured in different construction scenarios and with different shapes, including a cross-shaped fence with obstructions in a road construction scene, a cross-shaped fence without obstructions in an indoor construction scene, and a vertical fence with obstructions in a substation construction scene. Fig. 6b shows the ground truth of the fences obtained using the labelme annotation software [22], Fig. 6c displays preliminary predicted fence regions generated using our improved ENet segmentation network, and Fig. 6d shows the fence regions recovered using graphical processing algorithms.

<table>
<thead>
<tr>
<th>Table I</th>
<th>Table II</th>
<th>Table III</th>
</tr>
</thead>
<tbody>
<tr>
<td>improved ENet network models and the original ENet network on the fence dataset. It can be observed that the improved network model achieves overall better recognition performance without sacrificing recognition speed, demonstrating the effectiveness of the proposed network architecture and new activation function.</td>
<td>Table II presents the results of our proposed method when four different types of fences are identified. From Table 2, it can be observed that our proposed method achieves an accuracy of around 90% for fence identification in different scenes. Especially for data with occlusions, it seems that our method performs well in identifying fence boundaries. Additionally, it seems that our proposed method meets the processing speed requirements while achieving high accuracy throughout the entire identification process.</td>
<td>Table III summarizes the segmentation accuracy of our proposed method for the 540 images in the four different fence types. It can be observed that our proposed method demonstrates good robustness in segmentation, achieving an overall recognition rate of approximately 90%. This may meet the requirements for fence recognition in construction sites and provides a solid foundation for future work.</td>
</tr>
</tbody>
</table>

![Image](image_url)  

Figure 6. Experimental results of the method in this paper
TABLE I. EXPERIMENTAL PERFORMANCE OF DIFFERENT MODELS

<table>
<thead>
<tr>
<th>Model</th>
<th>mIoU</th>
<th>Average of speed(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENet</td>
<td>0.854</td>
<td>0.036</td>
</tr>
<tr>
<td>ENet+Leaky_ReLu6</td>
<td>0.863</td>
<td>0.035</td>
</tr>
<tr>
<td>Mul-scale ENet+CA+Leaky_ReLu6</td>
<td>0.896</td>
<td>0.038</td>
</tr>
</tbody>
</table>

TABLE II. EXAMPLE OF FENCE RECOGNITION

<table>
<thead>
<tr>
<th>Type</th>
<th>Origin Images</th>
<th>The Results of Our Method</th>
<th>IoU</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td>0.913</td>
<td>0.034s</td>
</tr>
<tr>
<td>B</td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
<td>0.887</td>
<td>0.037s</td>
</tr>
<tr>
<td>C</td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
<td>0.894</td>
<td>0.036s</td>
</tr>
<tr>
<td>D</td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
<td>0.885</td>
<td>0.039s</td>
</tr>
</tbody>
</table>

TABLE III. SEGMENTATION PERFORMANCE OF OUR METHOD

<table>
<thead>
<tr>
<th>Type</th>
<th>Number of Tests</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>116</td>
<td>0.925</td>
</tr>
<tr>
<td>B</td>
<td>147</td>
<td>0.894</td>
</tr>
<tr>
<td>C</td>
<td>143</td>
<td>0.912</td>
</tr>
<tr>
<td>D</td>
<td>134</td>
<td>0.883</td>
</tr>
</tbody>
</table>

V. CONCLUSIONS

This paper proposes a fence recognition method based on improved ENet neural network. The method utilizes the distinct color features of fences for neural network training and employs morphological algorithms to process the predicted images, enabling fast and accurate segmentation of the fence structures in construction site images. This method has the advantages of low cost, high accuracy, and low computational complexity. It can be used to develop safety guarantee systems for the stuff working at construction sites.

REFERENCES

SLPKT: A Novel Simulated Learning Process Model for Knowledge Tracing

Jingxia Zeng¹, Mianfan Chen¹, Jianing Liu¹, Yuncheng Jiang¹,²*
¹ School of Computer Science, South China Normal University, Guangzhou, China
² School of Artificial Intelligence, South China Normal University, Foshan, China
{2021023230, jiangyuncheng} @m.scnu.edu.cn

Abstract—Knowledge Tracing (KT) traces students’ changing knowledge states and predicts future performance based on their past performance. However, most existing methods underestimate the impact of students’ learning processes on performance prediction, and thus do not model the learning process well. To address this issue, we propose a novel Simulated Learning Process Model for Knowledge Tracing, which simulates the student’s learning process by reviewing historical performance and enhancing the role of related knowledge in prediction before solving exercises. We first use a state acquisition module to obtain the knowledge state. Then we mine important historical information to assist in solving the target exercise. Finally, a knowledge enhancement module is used to improve the knowledge prediction of the target exercise. Extensive experiments on four real-world datasets demonstrate that our method is effective and outperforms the state-of-the-art models.

Index Terms—Knowledge tracing, Learning process, Historical performance, Knowledge enhancement

I. INTRODUCTION

In recent years, the usage of various online education systems has increased significantly, and students generate a significant amount of learning data on these platforms and systems [3]. The primary research task for these learning data is to infer the student’s mastery level of knowledge based on these learning records, and then provide him with follow-up personalized services, such as learning path suggestions [6], course recommendations [5], and adaptive testing [18].

Knowledge Tracing (KT) [2] is an emerging research area in online learning, which uses the student’s past performance to predict the student’s future performance. In recent years, KT has gained widespread applications, drawing increasing attention from academia, and various methods for dealing with it have been proposed [4], [7], [9], [16], [19], [20].

Most existing KT methods employ Recurrent Neural Networks (RNNs) to process students’ learning records [4], [13], [21]. To predict student’s performance more accurately, there are some methods that have considered the influence of the student’s learning process. Existing methods for studying student’s learning process primarily suggest that student’s forgetting behavior and learning ability will influence final performance.

However, the existing methods are primarily restricted to the forgetting effect and the learning ability difference [10], [14], [15], which do not fully model student’s learning process, thus limiting the model’s predictive accuracy. In addition to the factors described above, we believe that the following learning process also affects the performance: After receiving a new exercise, students will go through a process of reviewing whether they have done similar problems, and then search for related knowledge and information to generate an idea to solve this new exercise. For example, as shown in Fig. 1, there has a student who is given a new exercise that involves the Formula for the Difference of Square (FDS). To complete this new exercise, first, he will review whether he has done exercises about the FDS before. If the student has solved similar exercises correctly or used the same concept (FDS) correctly to solve some exercises, the probability of the student solving this new exercise correctly should be higher, and vice versa. Furthermore, whether he did exercises related to the FDS or not, he will go through a process of searching for the FDS in his mind to solve this new exercise.

Therefore, we propose a novel Simulated Learning Process Model for KT (SLPKT), which can model the student’s learning process described above. Specifically, after the student get a new(target) exercise, we first use a knowledge state acquisition module to obtain the knowledge state. Then, to mine important historical information, we trace historical states based on the similarities between the target’s knowledge concept and the knowledge concepts of the historical exercises. Finally, a knowledge enhancement module is used to improve the prediction accuracy of the target exercise. Furthermore, we conduct a series of experiments on four public datasets and compare them to existing KT methods. Our results indicate that SLPKT outperforms existing KT methods in predicting student performance.

We summarize the key contributions of SLPKT below:

- To our knowledge, SLPKT is the first time that students’ learning processes are fully simulated to predict their performance after they get the new exercises.

Fig. 1. The learning process after students get a new exercise. In particular, e₁, e₄, and e₀ are exercises that involve the FDS.

DOI reference number: 10.18293/SEKE23-049
• We design a historical review module to retrieve historical important information to assist in solving the target exercise, and a related knowledge enhancement module that can enhance the role of knowledge related to the target exercise in predicting performance.
• Numerous experimental results on four public datasets show that SLPKT outperforms the state-of-the-art models, demonstrating the effectiveness of our method.

II. RELATED WORKS

The existing KT methods can be roughly divided into two types: traditional knowledge tracing and deep knowledge tracing. Since most of the current works are deep knowledge tracing, thus we mainly introduce deep knowledge tracing.

A. Traditional knowledge tracing

Bayesian Knowledge Tracing (BKT) [2] is a classic probability model on KT, which can be regarded as a special case of the hidden Markov model (HMM). Performance Factors Analysis (PFA) [12] and Learning Factors Analysis (LFA) [1] are essentially traditional psychometric models.

B. Deep knowledge tracing

Deep Knowledge Tracing (DKT) [13] introduces deep learning to KT for the first time. Augmenting Knowledge Tracing by Forgetting (DKT+Forgetting) [9] adds forgetting features to the DKT. Dynamic Key-Value Memory Network (DKVMN) [21] introduces memory enhancement neural networks into KT. To obtain knowledge state, Exercise-Enhanced sequential modeling for student performance prediction (EERNNA) [17] calculates the attention weights between hidden states by calculating the cosine similarity between the exercises. Context-aware Attentive Knowledge Tracing (AKT) [4] uses the same text feature extraction method as the EERNNA model, and further mines students’ guessing and mistaken behavior from the proposed semantic features. A Graph-based Interaction model for Knowledge Tracing (GIKT) [19] uses graph convolutional networks to capture exercise representations and knowledge concepts from the diagram of exercises, and uses a recap module to review relevant historical exercises to help students solve problems. Learning Process-consistent Knowledge Tracing (LPKT) [15] directly uses a learning-gain module and a forgetting module to model the student’s learning process to monitor the student’s knowledge state.

Although small parts of these methods modeled student’s learning process, they are all limited to the forgetting effect and learning ability difference. These methods do not take good account of the impact of student’s learning process. Therefore, we proposed SLPKT to simulate the learning process of students after they receive a new exercise to improve the performance and interpretability of the model.

III. PRELIMINARY

In an online education system, assuming there are the set of students $S = \{s_1, s_2, \ldots, s_J\}$, the set of exercises $E = \{e_1, e_2, \ldots, e_J\}$, and the set of knowledge concepts $L = \{I_1, I_2, \ldots, I_T\}$. Since each exercise is related to a specific knowledge concept, a dictionary is used to represent the relationship between the exercise and the concept, such as dictionary $\text{dict} = \{e_i : [I_{m_1}, \ldots, I_{m_n}], \ldots\}$, indicating that $e_i$ is related to $I_{m_1}, \ldots, I_{m_n}$ and so on.

In the KT task, students answer a series of exercises provided by the online learning platform sequentially, and the system will provide feedback on each exercise’s accuracy once the student has responded. Given an interaction sequence $X = \{x_1, x_2, \ldots, x_t\}$ and a new question $e_{t+1}$, where $x_t = (e_t, c_t, a_t)$, $e_t$ is the exercise, $c_t$ is the value corresponding to $e_t$ in the dictionary $\text{dict}$, and $a_t$ indicates whether the $e_t$ is answered correctly (1 means correct, 0 means false). The goal of KT is to monitor students’ changing knowledge state during the learning process and predict their performance at the next learning step $t + 1$.

IV. METHODOLOGY

In this section, we will introduce SLPKT in detail. As shown in Fig. 2, the model consists of four modules: (a) knowledge state acquisition module, (b) historical knowledge state review module, (c) related knowledge enhancement module, and (d) prediction module. Specifically, after the student is given a new exercise, the knowledge state acquisition module first obtains the knowledge state based on his past performance. Then the historical state review module reviews his historical performances to mine important historical information. Furthermore, the knowledge enhancement module enhances the role of related knowledge of this new exercise in predicting performance. Finally, we use the prediction module to predict student’s performance on this new exercise.

A. Knowledge State Acquisition

During the learning process, since student’s mastery of knowledge is constantly changing, we need to model the entire process of student learning to capture the changing knowledge state. For each learning step, we connect the exercise and the answer, and then project them into the $d$-dimension through linear transformations. So the input of the model is:

$$x_t = W_x^T [e_t \oplus a_t] + b_x,$$

where $\oplus$ denotes the vector concatenation operation, $W_x \in \mathbb{R}^{2d \times d}$ is the weight and $b_x$ is the bias. Then, we use LSTM to obtain the basic knowledge state $h_t$:

$$h_t = LSTM(x_t, h_{t-1}),$$

where $h_t$ and $h_{t-1}$ represent student’s knowledge states.

B. Historical Knowledge State Review

Whenever students are faced with a new problem, most of them first will review whether they have done similar exercises or used the same knowledge concept to solve some exercises. Thus, when completing the prediction task of KT, it should not only depend on the current knowledge state but also consider whether the historical knowledge states have reference significance for the current prediction work.
Therefore, we use an attention-based neural network model to calculate the similarities between the knowledge concepts involved in the target exercise and the knowledge concepts involved in all historical exercises. Then, according to these similarities, the corresponding historical knowledge states are integrated into the current knowledge state. In addition, we do not use the basic scaled dot attention mechanism because learning is temporary and memory declines.

Specifically, there are a key, query and value embedding layer that map inputs to keys, queries and values respectively. Let \( q_t \) and \( k_t \) represent the query and the key, which correspond to the knowledge concept involved in the exercise answered by the learner, and \( v_t \) is the value corresponding to the knowledge state at the learning step \( t \):

\[
q_t = K_\text{-}(\cdot)(c_t), \quad k_t = Q_\text{-}(\cdot)(c_t), \quad v_t = V_\text{-}(\cdot)(h_t), \quad (3)
\]

where \( c_t \) and \( h_t \) are the knowledge concept and the knowledge state respectively. And we use the softmax function to calculate the \( \alpha_{t,m} \) of the scaled dot product attention value:

\[
\alpha_{t+1,m} = \text{Softmax}\left(\frac{\exp(-\theta \cdot D(t+1,m))q_t^T k_m}{\sqrt{d}}\right), \quad (4)
\]

where \( k_m \) is the concept embedding through \( K_\text{-}(\cdot) \) at the learning step \( m \), and \( 1 \leq m \leq t \) means we depend on all the past learning steps. \( \theta > 0 \) is a learnable decay rate parameter and \( D(t+1,m) \) is temporal distance measure between time steps \( t+1 \) and \( m \).

Considering that a student may complete all the exercises many days apart, the relative distance of the exercises and the interval time are used to control the decay rate of the decay function simultaneously:

\[
it = \begin{cases} 
1, & \text{if the interval time} \geq 0, \\
0, & \text{otherwise}. 
\end{cases}
\quad (5)
\]

\[
D(t+1,m) = (|t+1-m| + \text{it}) \cdot \sum_{n=m+1}^{t+1} \gamma_{t+1,n}, \quad (6)
\]

\[
\gamma_{t+1,n} = \text{Softmax}\left(\frac{q_t^T k_n}{\sqrt{d}}\right), \quad \forall n \leq t+1.
\]

Then, according to the similarities of the calculated concepts, the corresponding historical knowledge states are integrated into the current knowledge state, and these important historical information is used to assist in problem-solving:

\[
h_t^{\text{hist}} = h_t + \sum_{m=1}^{t} (\alpha_{t+1,m} \cdot v_m). \quad (7)
\]

### C. Related Knowledge Enhancement

Since students’ mastery of each knowledge concept has its own independent evolution process, and students also will have a process of searching for related knowledge in their minds to solve the exercise after they get a new exercise. Therefore, to be able to better simulate this learning process, we need to enhance the role of knowledge mastery related to the target exercise on prediction.

To model the above behavior, we first calculate the correlation between the target exercise and the knowledge state, and the module’s inputs are the target exercise and the knowledge state, as shown in the Fig. 2 (c). Then, we normalize the correlation of each learning step with the correlation calculated by all previous learning steps:

\[
\alpha_t = h_t^{\text{hist}} W [e_{t+1} \oplus h_t^{\text{hist}} \oplus (e_{t+1} - h_t^{\text{hist}}) \oplus (e_{t+1} \cdot h_t^{\text{hist}})], \quad (8)
\]

\[
\alpha_{t+1,t}^2 = \text{Softmax}(\alpha_t),
\]

where \( e_{t+1} \) is the exercise at the learning step \( t+1 \) and \( h_t^{\text{hist}} \) is the knowledge state which has reviewed the historical information at the learning step \( t \). \( W \in \mathbb{R}^{d \times 4d} \), \( d \) is the dimension of \( e_{t+1} \)'s embedding vector.
Then, we put this correlation and knowledge state into the Gate Recurrent Unit (GRU) with an attentional update gate (AUGRU) for updating, which is a module transformed by DIEN [22].

\[
\begin{align*}
    \mathbf{u}_t &= \sigma(\mathbf{W}_u [\mathbf{h}_{t-1}^{hist}, \mathbf{h}_t] + b_u), \\
    \bar{\mathbf{u}}_t &= \mathbf{α}_{t+1} \cdot \mathbf{u}_t,
\end{align*}
\]

where \(\mathbf{u}_t\) is the update gate of GRU, \(\mathbf{W} \in \mathbb{R}^{d \times d}\) is the weight, \(b\) is the bias, \(\bar{\mathbf{u}}_t\) is the attention update gate and \(\mathbf{h}_t^{hist}\) is the hidden states of AUGRU. And \(\mathbf{h}_t\) is the final knowledge state generated by SLPKT:

\[
\mathbf{h}_t = (1 - \bar{\mathbf{u}}_t) \cdot \mathbf{h}_{t-1} + \bar{\mathbf{u}}_t \cdot \mathbf{h}_t^{hist}.
\]

\[ (10) \]

D. Prediction

In the student’s learning process, after the student is given a new exercise \(e_{t+1}\), he will get a knowledge state to solve this new exercise according to the above process. Therefore, we use the relevant knowledge state \(\mathbf{h}_t\) to infer the student’s performance on \(e_{t+1}\). We first connect the knowledge state \(\mathbf{h}_t\) and the exercise \(e_{t+1}\), and then project them to the output layer through a fully connected network activated by sigmoid:

\[
y_{t+1} = \sigma(\mathbf{W}_y \left[ \mathbf{e}_{t+1} \oplus \mathbf{h}_t \right] + \mathbf{b}_y) \in [0, 1],
\]

where \(\mathbf{W}_y\) is the weight and \(\mathbf{b}_y\) is the bias. Output \(y_{t+1}\) indicates the expected performance of the student in \(e_{t+1}\).

\[ (11) \]

### TABLE I

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ASSIST2009</th>
<th>ASSIST2012</th>
<th>ASSIST2017</th>
<th>EdNet1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students</td>
<td>3,352</td>
<td>27,485</td>
<td>1,709</td>
<td>5,000</td>
</tr>
<tr>
<td>Exercises</td>
<td>17,737</td>
<td>53,065</td>
<td>3,162</td>
<td>12,022</td>
</tr>
<tr>
<td>Concepts</td>
<td>123</td>
<td>265</td>
<td>102</td>
<td>142</td>
</tr>
<tr>
<td>Interactions</td>
<td>282,606</td>
<td>2,709,436</td>
<td>942,816</td>
<td>676,985</td>
</tr>
<tr>
<td>Avg.length</td>
<td>82.72</td>
<td>93.64</td>
<td>551.68</td>
<td>126.32</td>
</tr>
</tbody>
</table>

### V. EXPERIMENT

In this section, we conducted several experiments to investigate the effectiveness of SLPKT. First, we assess prediction error by comparing our model to other baselines on four common datasets. Then, we performed an ablation study on the historical knowledge state review module and the related knowledge enhancement module to demonstrate their effectiveness. Finally, we investigate the effect of different fixed lengths on our model and visualize the knowledge state obtained from each module to demonstrate that SLPKT has learned a more meaningful knowledge state.

A. Datasets

Four public datasets are used to evaluate the model’s validity which are commonly used in KT tasks. Table I shows the statistics for all datasets. We filter records without knowledge concepts. A brief description of all datasets is listed below:

ASSIST2009 comes from ASSISTments, an online tutoring system created in 2004. ASSIST2012 is collected from the same platform as ASSIST2009 during the school year 2012-2013. In this dataset, each exercise is only related to one knowledge concept, but one knowledge concept corresponds to several exercises. ASSIST2017 was utilized in the 2017ASSIST data mining competition. EdNet1 is a dataset of all student interactions with the system that Santa collected over two years.

B. Training Details

**Data Preprocessing.** We first sort all of the students’ learning records based on the learning steps of the answers, and then set all input sequences of the dataset to a fixed length of 200. For sequences shorter than a fixed length, it is filled to a fixed length with a vector of zeros. And student interactions with a sequence length of less than 3 are removed from all datasets.

**Training Settings.** We performed a standard 5-fold cross-validation on all models for all datasets. For each fold, 80% of the students are divided into the training set (60%) and the validation set (20%), and the remaining 20% as the test set. We randomly initialize all parameters in an even distribution, all hyperparameters are learned on the training set, and the test set is evaluated using the model that performs best on the validation set. Finally, we set some necessary parameters: The number of dimensions \(d\) mentioned in the text is 128, the batch size is 64, and the learning rate of the Adam algorithm is 0.002 for all trainable parameters. Finally, we select the area under the ROC curve (AUC) and accuracy (ACC) as evaluation indicators.

C. Baseline Methods

We compare the model with several previous methods. For a fair comparison, all of these methods are tuned for optimal performance.

- DKT [13] uses RNNs to assess student’s knowledge state, but we use LSTM in our implementation.
- DKT+ [9] adds forgetting features to the DKT, which are the interval time of the same exercises, the interval time of adjacent exercises, and the number of historical exercises of the target exercise.
- DKVMN [21] proposes a new dynamic key-value storage network model, which has a static matrix and a dynamic matrix to store and update the mastery of corresponding concepts respectively.
- SAKT [11] introduces the self-attention model to capture the correlation of relevant exercises from previous interactions to make predictions.
- AKT [4] uses two self-attention encoders to learn context-aware representations of exercises and answers, and it uses hyperparameters to retrieve knowledge gained in the past that is relevant to the current exercise.
- GIKT [19] uses a graph convolutional network to capture exercise representations from the relation graph of exercise and knowledge concepts, and uses a recap module to capture long-term dependencies.
- LPKT [15] is designed to complete KT tasks by simulating students’ learning and memory processes, and the model monitors students’ knowledge state by directly
modeling students’ learning processes using learning gain module and forgetting module.

- **KSGAN** [8] uses a Graph-Attention Network (GAT)-based model that leverages the knowledge structure between concepts and exercises to predict students’ performance.
- **CoKT** [7] retrieves the sequences of peer students who have similar question-answering experiences to obtain the inter-student information to make predictions.

### D. Student Performance Prediction

One of the most important metrics for evaluating the KT method is the experimental results of student’s performance prediction, so we conduct extensive experiments on all datasets to compare SLPKT to all baselines of student performance prediction and report the results of the five test folds in Table II. To provide robust evaluation results, performance was evaluated using AUC and ACC in all experiments. From Table II, we can see that SLPKT outperforms all other KT methods across all datasets and metrics, thus we believe that SLPKT is more aligned with students’ learning processes, resulting in more accurate predictions of their future performance. In addition, we note that on the ASSIST2012 dataset, the model is significantly better than the state-of-the-art LPKT model (AUC is improved by 2.6%), which indicates that the model has good adaptability to model the learning process of a large number of students.

#### E. Ablation Experiments

In this section, we conduct some ablation experiments to further show how each module in SLPKT affects final results. The prediction results of these variants are provided in Table III:

- **-HSRRNE** (Remove Historical State Review Module and Related Knowledge Enhancement Module): there is no simulation of the learning process after the students get a new exercise.
- **-HSR** (Remove Historical State Review Module): this variant model can only simulate the process of searching for related knowledge in student’s mind and enhance related knowledge in the prediction.
- **-RNE** (Remove Related Knowledge Enhancement Module): this variant model can only simulate the process of students reviewing the historical states to mine historical important information.

Table III shows that SLPKT is superior to all variant models, proving that the modules we added are practical. First, the role of the Related Knowledge Enhancement Module plays a crucial role in SLPKT, and if we removed it, it would lead to the greatest decline in the results. Then, reviewing the historical state in the KT task is more effective than not considering it. Finally, adding either of these two modules will be better than not adding them, which also shows that reasonable modeling of the student’s learning process will improve the accuracy of the prediction results.

#### F. Length Analysis and Knowledge State Visualization

To investigate the effect of different fixed lengths on our model, we evaluate our method’s performance on all datasets with four different lengths: 50, 100, 150 and 200, as shown...
in the Fig. 3. Shorter learning sequences of students often determine that the model cannot learn better performance. As the length increases, SLPKT can also maintain good performance.

To show SLPKT has obtained a more reasonable knowledge state, we visualize the knowledge state evolution process, as shown in Fig. 4. The probability in the knowledge state represents the student’s mastery of the current exercise and knowledge concept. In addition, correct answers are indicated by 1 and incorrect answers are indicated by 0. If the probability is closer to the actual answer, that means we have obtained a more accurate knowledge state. For the historical state review module, the state2 of e_3-e_12 obtain a more accurate knowledge state based on the historical information than state1. For the related knowledge enhancement module, the changes of state3 and state2 in e_3-e_5, e_8, and e_11-e_12 explain well that we have successfully enhanced the role of related knowledge.

VI. CONCLUSION

In this paper, we propose SLPKT by modeling the student learning process after they get a new exercise. Compared to the existing KT methods, we consider the students’ learning processes after they get a new exercise. To mine historical important information, we trace historical states based on the similarities between the target’s knowledge concept and the history exercises’ knowledge concepts. And a knowledge enhancement module is used to improve the role of the target exercise’s knowledge in performance predicting. We validated the performance of SLPKT on four public datasets and compared it to 9 excellent methods. Experimental results show that our method achieves better performance. In future work, we will further explore the better ways to simulate the learning process to improve the performance.

ACKNOWLEDGMENT

The works described in this paper are supported by The National Natural Science Foundation of China under Grant Nos. 61772210 and U1911201; The Project of Science and Technology in Guangzhou in China under Grant No. 202007040006.

REFERENCES

SARNet: A Self-Attention Embedded Residual Network for Multiclass Classification of Chronic Wounds

Chen Qian*, Boyin Yang† and Jiyun Li‡
School of Computer Science and Technology, Donghua University
Shanghai, China
Email: *chen.qian@dhu.edu.cn, †2212620@mail.dhu.edu.cn, ‡jyli@dhu.edu.cn

Abstract—Nowadays, chronic wounds have become an increasingly heavy healthcare burden. Therefore, wound classification is the most crucial task in wound diagnosis, which directly affects whether the treatment plan is optimal. This paper proposes a self-attention embedded residual network, or SARNet for short, which takes wound images as input and categorizes them into six types, i.e., burn wounds, surgical wounds, venous lower limb ulcers, pressure ulcers, diabetic foot ulcers, and normal skin. The classification accuracy of SARNet satisfactorily exceeds 80% mainly because its residual structure enhances the feature representation, and its built-in self-attention mechanism enables the global reference.

Keywords—chronic wound classification; residual network; polarized self-attention; multiclass classification.

I. INTRODUCTION

A chronic wound is defined as a breach in skin continuity that fails to achieve an anatomically and functionally intact state through an orderly and timely sequence of repair processes [1]. Traditionally, a patient’s wounds are manually analyzed, identified, and documented by the clinicians. However, the number of patients with chronic wounds is tremendous. Thus, performing large-scale wound classification by human means results in a massive medicare burden. Fortunately, with the rapid development of artificial intelligence (AI), we can now use computer vision technologies to classify wounds from only images.

Machine learning plays a vital role in computer vision with the aim of extracting critical information from images [3]. In the healthcare realm, it is frequently used to improve the quality and recognize the crucial features of medical imaging. However, conventional machine learning has certain limitations. For example, it requires that human experts remove unnecessary features before training, which is a time-consuming and laborious mission unsuitable for large-scale projects. Therefore, as an alternative, deep learning has received increasing attention [4]. By identifying and learning meaningful features from the totality of features by itself, deep learning can solve more complex problems with little human intervention.

Currently, most deep learning algorithms for chronic wound assessment are towards a binary classification, i.e., normal skin or physically harmed [5]. However, the sad fact is that such algorithms are useless in a real-world clinical diagnosis. Another sad fact is that the rest can barely provide a satisfactory classification in a multiclass fashion [6]. Thereby, we put forward a novel deep neural network that shows outstanding performance in the multiclass classification of chronic wounds. Based on the research achievement, we have successfully commercialized the proposed AI model and helped hundreds of patients to date.

II. RELATED WORK

In this paper, we only consider the most typical categories of chronic wounds [7]:

- Burn wounds (BW): Nearly 11 million people worldwide annually are severely burned, which requires long-time medical treatment.
- Surgical wounds (SW): Annually, roughly 4.5% of people worldwide undergo surgery that inflicts a wound.
- Venous lower limb ulcers (VLU): About 0.15% to 0.3% of people worldwide have a VLU.
- Pressure ulcers (PU): Each year, nearly 2.5 million people are suffered from PU.
- Diabetic foot ulcers (DFU): Approximately 34% of diabetic patients are threatened by DFU during their lifetimes, whilst more than 50% of patients with DFU become infected.

Most existing algorithms of wound classification aim to differentiate a wound from normal skin, among which the DFU diagnosis constitutes the majority. Because severe DFU usually leads to limb amputation, identifying DFU is in great demand so that treatment can come in time. Veredas et al. [8] proposed a hybrid system for automatic region segmentation and tissue recognition in an uncontrolled environment, which can detect the wound using color and texture features extracted by a multilayer neural network. Wannous et al. [9] implement color and texture region descriptors to perform a 3D-wound assessment. Wang et al. [10] used a cascaded two-level classifier to determine the boundaries of DFU. As image-based machine learning becomes even more sophisticated, more and more end-to-end models are adopted for better wound diagnosis.

However, the binary chronic wound classification is still considered ineffective in the real world. Hence, the multiclass
classification of chronic wounds has received increasing attention over the past several years. For example, Abubakar et al. [11] proposed a machine learning approach to distinguish BW or PU from normal skin, in which the image features are extracted by deep architecture, e.g., VGG-face, ResNet-101 or ResNet-152, and fed to an SVM classifier. Rostami et al. [6] put forward an integrated end-to-end DCNN classifier to divide the wound into multiple categories, including SW, DFU, and VLU. A total of 538 images of the natural wound are used in the experiment, which results in mean classification accuracy values of 94.28% for binary classification and 87.7% for ternary classification. Sarp et al. [12] performed a quaternary wound classification using a classifier model generated through interpretable artificial intelligence (XIA) and transfer learning, leading to an F1 mean score of 0.76.

Unfortunately, existing wound classification methods are generally unreliable. For example, SVM is frequently used to extract wound features [13]. Despite the experimental results that show accuracy improvements, they are barely convincing due to the small size of the evaluation set. Moreover, such experiments usually require specific lighting conditions (shading), markers, and skin colors. Otherwise, the model performs inadequately. Some other models adopt a strong contrast between binary and multi-class classifications, e.g., the accuracy drops rapidly from 97% to 72% [6]. For all these reasons, in this paper, we present SARNet, a self-attention embedded residual network using multi-branch topology, to classify the most common six types of chronic wounds precisely.

### III. OUR WORK

Currently, most research on chronic wound classification refers to a single type of wound, whereas the rest achieves low accuracy involving multiple types. We hereby present a self-attention embedded residual network, or SARNet for short, which is structured using a multi-branch topology. Notably, the branches adopt different convolution kernels, respectively, in order to obtain different receptive fields, thus thoroughly extracting features at various depths.

### A. Multi-branch Convolutional Neural Network

After VGG achieved a top-1 accuracy of over 70% on ImageNet classification [14], many innovations have emerged in making the network more complex to achieve high performance. For example, ResNet proposed a simplified dual-branch architecture that implicitly integrated many shallower models with the aim of training a multi-branch model to avoid the vanishing gradients problem [15]. Although complex neural architectures can generate networks with higher performance, the cost of computing resources or workforce becomes enormous. Moreover, some models are too sophisticated to be trained using ordinary GPUs, let alone the usage in practice. In spite of the inconvenience of implementation, complex models may reduce the parallelism and hence slow down the inference [16].

Our model is constructed on the basis of a simple VGG architecture. Since the wound surface usually appears as circular or irregular shapes, we adapt the multi-branch structure originally proposed in RepVGG [15] to enhance the representation of the network model. Each branch applies a specific receptive field and captures more relevant image features accordingly. Apparently, the residual branches are the key to SARNet architecture, which divides the training process into three paths. Each path contains downsampling and BatchNorm (BN) layers. The role of the BN layer is to normalize the data, which stabilizes the distribution of input data and thus accelerates the overall learning speed of the model.

The formulas used in the convolutional and BN layers are expressed as follows:

$$Conv(x) = \sum_i w_i x_i + b \quad (1)$$

$$BN(x) = \gamma \times \frac{(x - \text{mean})}{\sqrt{\text{var}}} + \beta \quad (2)$$

where $w$ is the weight of the convolutional kernel, $x$ is the input, $b$ is the bias, $\beta$ and $\gamma$ are the learnable parameters, $\text{mean}$ is the mean value and $\sqrt{\text{var}}$ is the variance. Equation 3 is obtained by substituting Eq. 1 into Eq. 2, which shows the convolutional layer with bias vectors obtained by fusing the BN with the previous convolutional layer. Using a multi-branch convolutional neural network with a parallel structure

![Fig. 1. An overview of SARNet.](image-url)
can improve the accuracy of the model during training and avoid the problem of vanishing gradients.

\[
BN(\text{Conv}(x)) = \frac{\gamma \times W(x)}{\sqrt{\text{var}}} + \frac{\gamma \times (b - \text{mean})}{\sqrt{\text{var}}} + \beta
\]  

(3)

**B. Polarized Self-attentive Mechanism**

Since convolution can only use local rather than global information to calculate the target pixel, this may introduce some bias. Suppose we treat each pixel in the feature map as a random variable and calculate the pairwise covariance between all pixels, i.e., the similarity between two random variables. In that case, we can enhance or weaken the value of each predicted pixel based on its similarity with other pixels in the image. Since the wound figure may appear at any position in the image (some wound figures may occupy a small proportion of the image with much irrelevant background), adding an attention mechanism between the fourth and fifth convolution layers can achieve global reference in the model training and prediction process, thereby enhancing the model classification performance.

In this paper, we use the polarized self-attentive (PSA) mechanism, which is realized as a combination of two branches. One performs a channel-wise self-attention mechanism, while the other is spatial-wise. Eventually, the results of the two branches are fused to generate the output of the structure, as demonstrated in Figure 2. To reduce the information loss caused by dimension reduction, PSA uses the polarized filtering mechanism, which maintains the size of \([H, W]\) in the spatial dimension and uses the size of \(C/2\) in the channel dimension. In addition, a non-linear function that regresses the output distribution at a fine granularity is used to enhance the information. In other words, the Softmax function is used to increase the attention range on the smallest tensor in the attention module, and then the Sigmoid function is used for dynamic mapping. Equations 4 and 5 calculate the weight of the channel branch and spatial branch, respectively.

\[
A^{\text{ch}}(X) = F_{SG}(W_{z}\theta_{1}(\sigma_{1}(W_{v}(X)) \times F_{SM}(\theta_{2}(W_{q}(X))))))
\]  

(4)

\[
A^{\text{sp}}(X) = F_{SG}(\sigma_{3}(F_{GP}(W_{q}(X)))) \times \theta_{t}(W_{v}(X)))
\]  

(5)

As a result, the fusion of the channel and spatial branches can be calculated as follows:

\[
PSA_{s}(x) = Z^{\text{sp}}(Z^{\text{ch}})
\]

\[= A^{\text{sp}}(A^{\text{ch}}(X) \circ^{\text{ch}} X) \circ^{\text{sp}} A^{\text{ch}}(X) \circ^{\text{ch}} X
\]  

(6)

**IV. Experiment**

**A. Dataset**

The dataset used in this paper is from the Kaggle big data competition platform. It contains 1777 wound images, including six categories: 323 images of BW, 209 images of SW, 447 images of VLU, 373 images of PU, 325 images of DFU, and 100 images of normal skin. These data are divided into a training set and a test set in a 7:3 ratio.

**B. Data Preprocessing**

All wound images are preprocessed via two steps. The first step is to augment the data by horizontal flipping, rotation, and random cropping, whereas the second is to process the wound images using mixup data enhancement [17]. Mixup is a data augmentation principle independent of data and a form of neighborhood risk minimization. It uses modeling between different categories to achieve data augmentation. First, two samples are randomly selected from the training samples for simple random weighted summation, and the labels of the samples are also correspondingly weighted and summed. Then, the predicted result and the label after weighted summation are used to calculate the loss and update the parameters in reverse differentiation. Mixup extends the training distribution by combining prior knowledge that linear interpolating feature vectors should lead to linear interpolation of relevant labels. In addition, the mixup method can reduce the considerable memory loss and sample sensitivity in the network and can reduce the memory of incorrect labels.

**C. Experimental Results**

To thoroughly investigate the classification performance, we use accuracy, precision, recall and F1-score to evaluate our SARNet.

We achieved an accuracy of 80.87% in the senary classification of the chronic wound using the SARNet. Heretofore, the highest record of senary classification is 75.64%, which is achieved by a VGG16 network using the AZH dataset [5].
So, we improved the accuracy by 5.23%. In addition, we also performed six quinary classifications on this dataset. The accuracy results are illustrated in Table I. It is worth noting that B, S, V, P, D, and N are the abbreviations of BW, SW, VLU, PU, DFU, and normal skin, respectively.

<table>
<thead>
<tr>
<th>Num of Classes</th>
<th>Classes</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 classes</td>
<td>BDNPS</td>
<td>84.24%</td>
</tr>
<tr>
<td></td>
<td>BDNPV</td>
<td>82.78%</td>
</tr>
<tr>
<td></td>
<td>BDNSV</td>
<td>83.45%</td>
</tr>
<tr>
<td></td>
<td>BDPSV</td>
<td>77.94%</td>
</tr>
<tr>
<td></td>
<td>BNPSV</td>
<td>83.99%</td>
</tr>
<tr>
<td></td>
<td>DNPSV</td>
<td>84.71%</td>
</tr>
<tr>
<td>6 classes</td>
<td>BDNPSV</td>
<td>80.87%</td>
</tr>
</tbody>
</table>

In addition, we compared the effects of mixup and attention mechanism on the senary classification, as shown in Table II, which provides ample evidence of their importance.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SARNet without mixup and attention</td>
<td>69.34%</td>
</tr>
<tr>
<td>SARNet without mixup</td>
<td>77.24%</td>
</tr>
<tr>
<td>SARNet without attention</td>
<td>71.92%</td>
</tr>
<tr>
<td>SARNet</td>
<td>80.87%</td>
</tr>
</tbody>
</table>

Fig. 3. The performance of SARNet.

To further demonstrate the effectiveness of this network model for six classifications in the chronic wound image dataset, we calculated the model’s accuracy, precision, recall, and F1-score, as shown in Figure 3. It can be seen that the precision, recall, and F1-score are also around 80% for all wound images, except for normal skin.

V. CONCLUSION

A high-performance classifier is urgently needed to classify wounds with less financial and time costs. This paper presents SARNet, which is constructed based on a multi-branch topology that enables various convolutional kernels referring to receptive fields at different levels of granularity for thorough feature extraction. Additionally, SARNet is introduced by a polarized self-attentive mechanism to capture long-distance dependencies for better global reference. The experiment shows SARNet outperforms all other existing approaches.

REFERENCES

**AT-NCF: An Attention-based Time-aware Neural Collaborative Filtering Approach for Personalized Recommendation**

**1st Yimin Peng**  
Key Laboratory of Knowledge Processing &Networked Manufacturing  
Hunan University of Sci & Tec  
Xiangtan, China  
yiminp@yeah.net

**2nd Rong Hu**  
Key Laboratory of Knowledge Processing &Networked Manufacturing  
Hunan University of Sci & Tec  
Xiangtan, China  
ronghai@126.com

**3rd Yiping Wen**  
Key Laboratory of Knowledge Processing &Networked Manufacturing  
Hunan University of Sci & Tec  
Xiangtan, China  
ypwen81@gmail.com

**Abstract—** With the explosive growth of data, how to apply these data with time characteristics to recommendation is particularly important. In order to do that, an attention-based time-aware neural collaborative filtering approach named AT-NCF is proposed in this paper. Concretely, short-term preferences of users are learned by a time perception network, and long-term preferences part are modeled by a historical interaction matrix. Then, the short-term preferences are concatenated with long-term preferences as the dynamic user preference vector. After that, high-order user-item feature interactions are learned by a general neural collaborative filtering framework which includes two types of DL models, Deep Matrix Factorization (DMF) and Multiple-Layer Perception (MLP). Finally, the predicted scores are output from the final layer of AT-NCF. Experimental results on the real-world e-commerce dataset verify the effectiveness of the proposed method.

**Keywords—** time-aware, attention mechanism, user preference modeling, neural collaborative filtering, deep learning

I. INTRODUCTION

With the spread of the Internet, the world has entered a new era of information overload and abundance [1]. As an efficient implement to alleviate the difficulty, personalized recommendation system is widely used to supply personalized information service for various business and fields [2].

The current recommendation algorithm is based on several methods, such as Collaborative Filtering (CF) [3][4], Matrix Factorization (MF) [5][6], Deep Learning (DL) [7] methods and Neural Collaborative Filtering (NCF) [8][9][10]. The CF-based methods firstly map users and items to latent factors which share the same latent space, and then use a linear function, such as inner product or cosine distance, to predict the user’s preference for items [11]. The MF algorithm use a latent feature vector to connect each user to the item. The DL-based models generally use neural networks to extract abstract feature representations of users and items or to learn the matching score [12]. The NCF combine nonlinear and linear models.

At present, existing NCF methods generally use historical interactions between users and items to build a user preference model or item similarity model without considering the interaction time [13]. Therefore, we propose an AT-NCF approach that takes interaction time into account in user preference modeling. Concretely, AT-NCF contains two parallel processes and a final fusion process: user preference modeling based on time-aware attention network, matching score prediction based on the fusion of two parts. The main contributions of our works are as follows:

- We use a time-aware attention works to model the dynamic user preference, so as to obtain optimized input embedding for the training of the DL model.
- We use neural collaborative filtering layer to combine DMF and MLP, to model the multidimensional user-item feature interactions to get the prediction scores.
- We performed experiments with different dimensions on an e-commerce dataset, to demonstrate the validity and rationality of the proposed AT-NCF method.

II. RELATED WORK

In this part, we state the application of the attention mechanism and time factor and DL-based model in CF recommendations system, respectively.

A. Collaborative Filtering based on Deep Learning

In deep learning-based recommendation work, Deep Belief Network (DBM) [14] is the earliest neural network model used in recommendation systems. A hierarchical DL model called Collaborative Deep Learning (CDL) [15], which collectively performs deep representation learning based on content and CF recommendation based on the score feedback matrix. Gradually, hybrid recommendation models with both representational learning and deep learning capabilities have become the focus of research. The NeuMF is a specific method under the NCF [7] framework that explains in detail how to learn NCF which emphasizes the binary attribute of implicit data using probabilistic model. ConvNCF [9] which element cross products is used to explicitly model pairwise correspondences between embedded spatial aspect, and convolutional neural network is used to learn the higher order correspondence. Deep Collaborative Filtering (DeepCF) [2] which combining the advantages of presentation learning and matching equation learning, a deep learning network is built to predict the user-item interaction matrix.

B. Collaborative Filtering based on Attention

Recently, the attention mechanism [16] has been widely used in various recommendation systems. For example, Attention CF (ACF) proposed by Chen et al. [17] in 2017 is the first CF recommendation model based on the attention network. ACF introduces two types cases: 1) item-level attention; 2) component-level attention. The former assumes that different historical interactive items have different contributions to user preference modeling and assign different weights to historical interactive items. Then, ATRank proposed by Zhou et al. [18] in 2018 models user preference by considering various heterogeneous behaviors. NAIS proposed by He et al. [8] is an item-based CF method that uses neural networks to model item similarity.
C. Collaborative Filtering based on Time-aware

The time-aware recommendation model is an intuitive and effective method for modeling time-series information. These methods generally adopt the recommendation algorithm based on neighborhood, which pays more attention to the observation items closest to the current moment. TimeSVD++ proposed by Koren et al. [19] is a well-known time-aware recommendation model that improves SVD++ by introducing time-varying biases for each user and item at each time step. Wang et al. [20] proposed an attention model based on time perception. Absolute time signals are used to represent users’ periodic behaviors, and relative time signals are used to represent the time relationship between items. TiSASRec [21] directly modeled the interaction timestamp to learn the influence of different time intervals on the next prediction.

III. THE PROPOSED METHOD

A. General Framework

The general framework of the proposed AT-NCF model is shown in Fig. 1, and the model works as follows: (1) The input layer: this includes the user ID, the item ID, and the user-item interaction history. (2) The embedding layer: the user ID, the item ID are encoded as one-hot vectors, and the user-item interaction history are encoded as multi-hot vectors. (3) The attention layer: the network is used to get individual weights, for each recent historical interaction item with which the user interacts (4) The prediction layer: DMF part, the linear user-item feature interactions are representation by the inner product of the user and the item. MLP part, the user and item embedding vector are taken as the input, and the high-order user-item feature interaction is learned through the multi-layer neural network. (5) The output layer: here, AT-NCF predicted the user-item matching score by integrating the interaction vectors of the two models.

B. Attention-based Dynamic User Preference

Generally, users’ preferences are an organic combination of historical and recent preferences. Therefore, we learn the contribution weight of each of the nearest k interaction terms through the attention network. The optimal recent-k value, we get it through experiments. The contribution weight of each interaction item to recent user preference modeling is learned through the attention network, as follows:

\[ a(j) = W_2^T \phi(W_{11} e_i^p + W_{12} e_j^p + W_{13} e_f^p + b_1) + b_2 \]

where \( a(j) \) is the resulting importance weight of the historical interacted item \( j \); \( W \) and \( b_k \) are weight matrices and biases of the attention network, respectively; \( \phi(x) \) is the ReLU. The embedding vector weighted sum of recent interactive items is used as short-term user preferences, and the long-term user preferences are modeled as the arithmetic mean of user historical interactive items, as shown below:

\[ P_u^1 = \frac{1}{|R(u)|} \sum_{j \in R(u)} e_j^p \]

where \( R(u) \) is the item that user \( u \) has interacted with in the former time, and \( |R(u)| \) is the numeral of items in \( R(u) \). The dynamic user preference vector is modeled as a combination of short-term and long-term preferences, as shown below:

\[ P_u = \alpha \cdot P_u^1 + (1 - \alpha) \cdot P_u^2 \]

C. DL-based feature interaction learning

Deep learning has been shown to approximate any continuous function, it’s very suitable for learning complex matching purpose. The DMF interaction vectors and the MLP interaction vectors are fused in the output layer of AT-NCF for matching score prediction.

Since dynamic user preferences \( P_u \) and item embedding \( e_i^p \) as a given variable, two separate networks are used to learn the hidden representation vectors of users and items as follows:

\[ U_u = M_{l_1} \phi(M_{l_2} P_u + c_1) + c_1 \]

\[ V_i = N_{l_2} \phi(N_{l_1} e_i^p + d_1) + d_1 \]

where \( l_1 \) and \( l_2 \) are the layer numbers of the user model and the item model, respectively; \( M/N \) and \( c/d \) are the weight matrices and biases of the user/item model, respectively. The linear feature interaction between the user and the item is obtained by the inner product, as follows:

\[ y_{ui}^{MLP} = (H_u U_u) \phi(H_i V_i) \]

where \( \phi \) is the inner product between two vectors; \( H_u \) and \( H_i \) are mapping matrices. Meanwhile, we also use MLP to learn nonlinear high-order feature interactions, as shown below:

\[ y_{ui}^{MLP} = L_{l_1} \phi(L_{l_2} [P_u, e_i^p] + g_1) + g_1 \]

where \( l \) is the layer number of the model, \( [P_u, e_i^p] \) is the vector concatenation of the dynamic user preference vector and the item embedding vector. \( L \) and \( g \) are the weight matrices and biases, respectively.
D. Fusion and Learning

Since the model takes 2 ways to model users and items, it is a common practice to combine the features of the 2 paths by concatenation, which has been widely used in multimodal deep learning. To do this, we concatenate the prediction vectors of the two DL models into a fully connected layer to learn the user-item matching score. Therefore, the final prediction layer is created by concatenating the two models with a final hidden layer, it is formulated as follows:

\[ \hat{r}_{ui} = \sigma(W_{out}[\beta \cdot y_{DMF} + (1 - \beta) \cdot y_{MLP}]) \]  

(9)

where, \(W_{out}\) is the output weight matrix of the model, \(\sigma\) is the sigmoid activation function. Our model is a CF model based on implicit feedback information, so the binary cross-entropy loss function is adopted as the objective function as follows:

\[ L(\theta) = \sum_{(u,i) \in R^+ - u_i} \log r_{ui} + (1 - r_{ui})(1 - \log \bar{r}_{ui}) \]  

(10)

where \(R^+\) and \(R^-\) denote positive and negative sample sets, respectively; \(r_{ui}\) is the real feedback, \(\hat{r}_{ui}\) is the predicted matching score, \(\theta\) is the hyperparameter set of the model.

IV. EXPERIMENTS

In this section, we introduce the experimental designs of this experiment and answer the following questions:

- **RQ1.** Does our proposed AT-NCF model achieve better performance than the correlation recommended methods?
- **RQ2.** Whether the key operations we proposed in this work improve the recommendation performance?
- **RQ3.** How do the hyperparameters of the AT-NCF model relate to the recommended performance?

A. Experimental Designs

<table>
<thead>
<tr>
<th>Field</th>
<th>Taobao</th>
<th>Taobao-mini</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>987,994</td>
<td>4,856</td>
</tr>
<tr>
<td>Items</td>
<td>4,162,024</td>
<td>242,329</td>
</tr>
<tr>
<td>Categories</td>
<td>9,439</td>
<td>5,070</td>
</tr>
<tr>
<td>Actions</td>
<td>100,15M</td>
<td>0.5M</td>
</tr>
<tr>
<td>Avg. actions/user</td>
<td>101.37</td>
<td>102.97</td>
</tr>
<tr>
<td>Avg. add-to-favor/user</td>
<td>2.92</td>
<td>2.67</td>
</tr>
<tr>
<td>Avg. add-to-carts/user</td>
<td>5.60</td>
<td>5.66</td>
</tr>
<tr>
<td>Avg. purchases/user</td>
<td>2.04</td>
<td>2.10</td>
</tr>
</tbody>
</table>

Datasets. This experiment selected the e-commerce dataset provided by Alibaba Group: Taobao user behavior data. The dataset format is like MovieLens-20M, with each row representing a user interaction with an item, and each column representing the user ID, item ID, item-category ID, interaction type (click, collection, shopping cart and purchase), and timestamp. The details are summarized in Table I.

Evaluation Metrics. To evaluate the effectiveness of our method, we used the leave-one-out evaluation method used in this experiment. For each user, the most recent interaction is taken as the test set, and the remaining data is taken as the training set. In the ranking of test items, this experiment follows a general strategy, that is, to randomly sample 100 items without user interaction and rank the test items among them. In addition, the performance of the ranked list is judged by the Hit Rate (HR) and the Normalized Discounted Cumulative Gain (NDCG).

Baselines. To verify the effectiveness of our method, we compared the performance of the following methods:

- **Item-KNN:** Typical item-based CF algorithm. This experiment follows the settings in the reference so that it can be applied to implicit data.
- **BPR-MF** [22]: Bayesian personalized ranking is a ranking algorithm based on matrix decomposition, uses ranking loss to decompose the user-item interaction matrix.
- **Deep-CF** [2]: Combine the CF based, on presentation and on function learning. The implicit feedback information is used as the input of two different types of DL models.
- **Neural-CF** [7]: A general neural CF model with DL-based matching function learning, which combines hidden layers of GMF and MLP to learn the interaction function.
- **NAIS** [8]: A two-layer neural collaborative filtering method with item similarity information.
- **TiSASRec** [21]: It is a state-of-the-art time-aware model, it explicitly models the timestamps of interactions.

Parameter Settings. The comparison methods use the parameter settings given in the original literature. Our model is set up using the similar strategy.

B. Performance comparison (RQ1)

We record the convergence process of each model through experiments, Fig. 2 shows the performance of the seven methods in HR@10 and NDCG@10 for 20 iterations. Apparently, DL-based models (NCF and NAIS) are superior to similarity-based method and MF-based models (Item-KNN and BPR-MF), indicating that the DL models have capabilities in high-order representation modeling and nonlinear feature interaction modeling. The time-aware recommendation method (TiSASRec) performs better than the DL-based recommendation method (DeepCF) by considering the interaction time-stamp information in user preference modeling. Overall, we can see that AT-NCF achieved the best performance. Compared with the base model (Deep-CF), it also leads by nearly 5% in both indicators. This shows that the attention mechanism and time-aware modeling has a great effect for recommendation performance.

C. Effectiveness of dynamic preference modeling (RQ2)

Due to the dynamic user preference modeling based on attention time-aware network is the key operation in ATNCF, we conduct experiments to demonstrate its effectiveness in this section. We set the variables of the experiment as 1) Do not use the attention network, named AT-NCF-un-a; 2) Use the attention network but not the time factor, named AT-NCF-un-t. Then, we compared the performance of several related models. The details of the above experiments are shown in Table II. The experimental results show that the method using time factor is better than the method without time factor among all the methods using attention mechanism. At the same time, it shows that interaction time information plays an important role in modeling user preferences and accurate top-k recommendations. In general, the experimental results demonstrate the advantages of time factor embedding and attention mechanism combination.

<table>
<thead>
<tr>
<th>Field</th>
<th>Taobao</th>
<th>Taobao-mini</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>987,994</td>
<td>4,856</td>
</tr>
<tr>
<td>Items</td>
<td>4,162,024</td>
<td>242,329</td>
</tr>
<tr>
<td>Categories</td>
<td>9,439</td>
<td>5,070</td>
</tr>
<tr>
<td>Actions</td>
<td>100,15M</td>
<td>0.5M</td>
</tr>
<tr>
<td>Avg. actions/user</td>
<td>101.37</td>
<td>102.97</td>
</tr>
<tr>
<td>Avg. add-to-favor/user</td>
<td>2.92</td>
<td>2.67</td>
</tr>
<tr>
<td>Avg. add-to-carts/user</td>
<td>5.60</td>
<td>5.66</td>
</tr>
<tr>
<td>Avg. purchases/user</td>
<td>2.04</td>
<td>2.10</td>
</tr>
</tbody>
</table>

Fig. 2. Performance of HR@10 and NDCG@10 on Taobao-mini.

TABLE II. EFFECTS OF KEY OPERATIONS ON THE MODEL

<table>
<thead>
<tr>
<th>Method</th>
<th>Hit Rate</th>
<th>NDCG</th>
<th>Avg. Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neu-CF</td>
<td>0.4257</td>
<td>0.3516</td>
<td>6</td>
</tr>
<tr>
<td>Deep-CF</td>
<td>0.4613</td>
<td>0.3652</td>
<td>5</td>
</tr>
<tr>
<td>NAIS</td>
<td>0.4739</td>
<td>0.3718</td>
<td>3</td>
</tr>
<tr>
<td>AT-NCF-un-a</td>
<td>0.4716</td>
<td>0.3693</td>
<td>4</td>
</tr>
<tr>
<td>AT-NCF-un-t</td>
<td>0.4964</td>
<td>0.3928</td>
<td>2</td>
</tr>
<tr>
<td>AT-NCF (both have)</td>
<td>0.5172 ▲</td>
<td>0.4169 ▲</td>
<td>1</td>
</tr>
</tbody>
</table>
D. Sensitivity Analysis of Hyperparameters (RQ3)

The control variable method was used in all the following experiments. Relevant parameters include [vector/model fusion coefficient, the number of latent factors/recent-k items]. From Figure 3 (I), we can see that the fusion coefficient in dynamic user modeling has a great impact on the model performance, because accurate user modeling is the key to high-quality recommendation. When the fusion coefficient is set to 0.5, the model gets the best performance, which shows that short- and long-term preferences are equally important in our model. Fig.3 (II) shows the ranking gets better and better as the number of factors increases. From Fig.4 (I), that 0.5 is the most appropriate coefficient, indicating that model based on linear method and the DL method are equally important in the integration model. Finally is the fusion coefficient of the two models, from Fig.4 (II), in modeling short-term user preferences, the results in the figure indicate that the recommended performance of the last 10 interactive items is a better value.

Fig. 3. Effects of fusion coefficient of the two vectors, and effects of the number of latent factors.

Fig. 4. Effects of fusion coefficient of the two models, and effects of the number of recently interacted items.

V. CONCLUSION AND FUTURE WORK

In this work, we proposed an attention-based time-aware multi-layer NCF recommendation model, AT-NCF, for personalized recommendation. Specifically, long- and short-term interests of users are combined to construct the finally user preference model. For matching function learning, we use DMF and MLP to learn the linear and nonlinear user-item feature interaction. In the final fusion layer of our model, the feature interaction vectors of the two models are fused to obtain the final prediction score. Extensive experiments show that our method outperforms the existing CF methods.

In the future, our method can be further studied. Actually, in the real recommended scenario, there is a lot of explicit data, social network data, and other related auxiliary information available. That is one of the focuses of our future work.

ACKNOWLEDGMENT

This work was supported by the Natural Science Foundation of China (No. 61702181,62177014), and the Research Foundation of Hunan Provincial Education Department of China (No. 20B222).

REFERENCES


MB-DP: A Multi-behavior Recommendation Model Integrating Dynamic Preferences

Cairong Yan, Xiaopeng Guan, Haixia Han, Zhaohui Zhang
School of Computer Science and Technology, Donghua University
{cryan, zhzhang}@dhu.edu.cn, {xpguan, haixiahan}@mail.dhu.edu.cn

Abstract—Multi-behavior recommendation has gained significant attention in recent years for its ability to outperform single-behavior models. Current research related to multi-behavior models leaves room for improvement in the following two areas. First, the noise carried by individual behaviors and the additional noise generated during behavior processing is often overlooked, and these can ultimately degrade recommendation performance. Second, the specific time period of behavioral interactions and the frequency of interactions within that time period are also not taken into account. To address the above limitations, we propose a multi-behavior recommendation model integrating dynamic preferences (MB-DP) that captures dynamic interests while smoothing and denoising multi-behavior information. MB-DP extracts low and high-order semantics from various behaviors and unifies the measurements to generate interaction predictions. Additionally, it analyzes the interaction time and frequency of each behavior using gated recurrent units to capture the dynamic preferences of users and improve the prediction values. Extensive experimental results on two real-world datasets show that MB-DP significantly improves recommendation performance compared to the state-of-the-art baselines.

Keywords—Recommender System; Multi-behavior; Dynamic Preference; Graph Neural Network; Gated Recurrent Unit

I. INTRODUCTION

With the explosive growth of information, users struggle to make decisions among countless options [1]. Recommender systems based on user behavior modeling are a good solution to this problem in various fields.

The use of behavioral data has increased in fields such as e-commerce and social networks [2]. However, single interaction types inadequately represent user interests, as interactions are often multi-typed. Multi-behavior recommendation has become a key research direction in recommender systems. Current models include deep learning-based and graph neural network-based approaches. Deep learning-based models consider the cascading relationship between behavior types and use multi-task learning for optimization [3]. Graph neural network-based models, like MB-GCN and GHCF, mine high-order interaction information between users and items, providing diverse content and structural information [4, 5, 6].

Multi-behavior recommendation models face two key issues: 1) existing models don’t address noise impact in multi-behavior information on recommendation performance, leading to over-smoothing and affecting personalization [7]; 2) existing models do not consider users’ dynamic preference information. As shown in Figure 1, in a multi-behavior recommendation model that does not consider dynamic preferences, two users have the same interaction behavior type for \( i_1 \), so similar preference degrees will be judged. However, by considering dynamic preferences over three time periods, we can see that only user \( u_1 \) interacts with item \( i_1 \) in the third period, and \( u_1 \) has more interactions with \( i_1 \) in the first period, indicating that \( u_1 \)'s preference for \( i_1 \) is stronger than \( u_2 \)'s. In real-world scenarios, users’ preferences [8] and item popularity [9] can change over time, and capturing changing preferences requires factoring in time to capture dynamic preferences from users’ multi-behavior.

Therefore, to address the aforementioned problems, in this paper, we propose a novel multi-behavior recommendation model integrating dynamic preferences (MB-DP). The goal of MB-DP is to establish a recommender system model based on multi-behavior learning and optimized prediction values using dynamic preferences. Specifically, We learn meta-knowledge from behavior interaction patterns, obtaining low and high-order personalized semantics. To reduce noise, we normalize initial measurements and use multi-layer graph neural networks. MB-DP analyzes time periods and interaction frequency to capture users’ dynamic preferences and optimize recommendation results. The contributions of our paper are as below:

(1) We propose MB-DP, a new recommendation network that considers multi-behavior patterns and dynamic preferences, and extracts personalized semantics while smoothing out noise from multiple behaviors.

This work is supported by Shanghai Science and Technology Innovation Action Plan Project (No. 22511100700)
DOI Reference Number: 10.18293/SEKE23-130

Figure 1. User preferences relate to multi-behavior types. A time-dynamic multi-behavior modeling approach captures more information.
(2) We use gated recurrent units to analyze interaction time and frequency, and correct multi-behavior learning results towards optimized prediction values.

(3) We show the effectiveness of MB-DP on two real-world datasets and conduct ablation studies to understand the model design. MB-DP outperforms baselines from various studies in terms of recommendation performance.

The paper is structured as follows: Section 2 reviews related work; Section 3 defines the research problem; Section 4 details our MB-DP model; Section 5 presents experimental results; Section 6 concludes our work and discusses future research.

II. RELATED WORK

The use of graph learning methods can uncover higher-order connectivity between users and items, with the core idea of enhancing node representation using (higher-order) neighborhoods [10]. NGCF [11] employs multi-layer graph neural networks for higher-order information aggregation, while LightGRCN [12] preserves user and item embeddings to prevent over-smoothing. Zhang et al. proposed a new second-order continuous GNN with better interpretability and avoided over-smoothing [7]. GAML [13] captures potential correlations between different scale graph knowledge, while InstantGNN [14] proposes an incremental calculation method for graph representation matrices to address continuous changes in large-scale dynamic graphs.

Multi-behavior recommendation models treat added behaviors as auxiliary compared to single-behavior models. NMTR [3] shares user and item embedding layers across behavior types. MB-GCN [5] unifies user-item interaction matrices into a graph, capturing behavior strength and semantics through propagation layers. MB-GMN [6] uses meta-learning to incorporate multiple behavior patterns and learn behavior representations related to behavior types. GHCF [4] jointly embeds node representations and relationships into multirelational prediction using GCN and performs non-sampling optimization under multi-task learning. S-MBRec [15] performs GCN for each behavior, differentiating the importance of different behaviors through a supervised task, and capturing the commonality of embeddings between target and auxiliary behaviors with a star-shaped contrastive learning task. However, existing multi-behavior recommendation models often overlook the positive impact of users’ dynamic preferences on recommendations. MB-DP addresses this issue by designing a dynamic preference module.

Although the above multi-behavior recommendation models consider the impact of multiple behaviors on personalized recommendations, they do not consider the important time factor in real-world scenarios, where customer preferences for items change over time [16]. SASRec [17] identifies relevant items from user history to predict the next item in each time period. FISSA [18] improves SASRec by incorporating a global representation learning module that balances local and global representations using candidate item information. DUMN [19] designs a user memory network to model long-term interests in a granular way. Considering the influence of time length, URPI-GRU [20] learns users’ short-term preferences through a GRU model and uses K-nearest neighbor sequence mining to explore users’ long-term preferences. However, these models typically do not sufficiently extract high-order content and structural information from interactions.

III. PROBLEM DEFINITION

We represent the set of users and items with \( U = \{ u_1, u_2, \ldots, u_M \} \) and \( I = \{ i_1, i_2, \ldots, i_N \} \), respectively. To capture multi-behavior interactions, we use a three-dimensional matrix \( X \in R^{K \times M \times N} \) where \( x_{uk} = 1 \) if there is an interaction of the \( k \)-th behavior category between user \( u \) and item \( i \), and 0 otherwise. To capture temporal information, we use a four-dimensional matrix \( T \in R^{M \times N \times D \times K} \) where \( t_{ukd} = z \) represents the number of interactions of the \( k \)-th behavior category between user \( u \) and item \( i \) during the \( d \)-th time period. We formulate the problem as predicting the probability of user \( u \) performing the target behavior on item \( i \) using \( X \) and \( T \) as inputs.

IV. METHODOLOGY

We now introduce the details of our proposed MB-DP model, as shown in Figure 2. It consists of three core modules.

A. Behavior Semantic Extraction Module

Different behaviors represent various user preferences, making some irrelevant interactions act as noise. Previous multi-behavior models focus on preference differences, while our goal is to extract user-specific behavior semantics.

1) Low-order Personalized Semantic Extraction

We use a method inspired by MB-GMN [6] to learn meta-knowledge in multi-behavior interaction patterns and obtain personalized user and item embeddings with behavior semantic information. To obtain the initial contextualized user embedding \( \hat{R}_u \in R^{M \times 3 \text{dim}} \), we concatenate the behavior embedding \( E^b_u \in R^{M \times \text{dim}} \) of the user under behavior \( b \), the user embedding \( E_u \in R^{M \times \text{dim}} \), and the project interaction information embedding \( E^u_{i, N} \) in the second dimension:

\[
\hat{R}_u = E^b_u \oplus E_u \oplus E^u_{i, N},
\]
where $E_{u,k}^k \in \mathbb{R}^{M \times 2dim}$ is obtained by normalizing the interaction matrix $R^k \in \mathbb{R}^{M \times N}$ and performing a cross item with the project embedding $E_i \in \mathbb{R}^{N \times 2dim}$.

To balance the degree of information acquisition of the contextualized user embedding and the user embedding $E_u$ under behavior $k$, we pass the initial contextualized user embedding $H_{u,k}^k$ through a linear layer with an input length of $3dim$, an output length of $dim$, and apply LeakyReLU for non-linear activation. This yields the contextually balanced user embedding $H_{u}^k \in \mathbb{R}^{M \times dim}$:

$$H_{u}^k = \text{LeakyReLU} \left( \text{MLP} \left( H_{u,k}^k \right) \right).$$

(2)

To address the problem of low efficiency in meta-learning caused by high dimensionality, we use low-rank decomposition to transform meta-learning by learning two low-rank projections, $\hat{V}_u^{k,1} \in \mathbb{R}^{M \times dim \times \text{rank}}$ and $\hat{V}_u^{k,2} \in \mathbb{R}^{M \times \text{rank} \times dim}$ for behavior $k$, and obtain the contextualized user embedding $E_{u,k}^k$ as follows:

$$E_{u,k}^k = \text{operate} \left( \text{operate} \left( E_u, \hat{V}_u^{k,1} \right)^k, \hat{V}_u^{k,2} \right),$$

$$\text{operate}(a, b) = \sum(\text{unsqueeze}(a) \odot b).$$

(3)

Here, $\hat{V}_u^{k,1}$ and $\hat{V}_u^{k,2}$ are obtained by applying a Linear layer with input length $dim$ and output length $\text{rank} \times dim$ to the contextually embedded user embedding $H_{u,k}^k$ after information balancing, and then reshaping the resulting tensor into $R^{M \times \text{rank} \times \text{dim}}$ and $R^{M \times \text{rank} \times \text{dim}}$ respectively.

By concatenating $E_{u,k}^k$ with $E_u$, we obtain the user embedding $E_{u}^k \in \mathbb{R}^{M \times 2dim}$ with personalized behavioral semantic information.

2) High-order Connected Semantic Extraction Module

We use graph neural networks to capture the similarity between users or items influenced by different behaviors.

Under behavior $k$, we define the adjacency matrix $A^k \in \mathbb{R}^{(M+N) \times (M+N)}$ as follows:

$$A^k = \begin{pmatrix} 0 & X^k \\ X^k \end{pmatrix}.$$  

(4)

We use the symmetric normalization matrix $\hat{A}^k \in \mathbb{R}^{(M+N) \times (M+N)}$ to smooth the input matrix that needs to be processed, and the symmetric normalization matrix is defined as follows:

$$\hat{A}^k = D_k^{-1/2} A^k D_k^{-1/2}.$$  

(5)

Here, $D_k$ is a diagonal matrix of size $(M+N) \times (M+N)$, and the values on the diagonal of $D_k$ represent the number of non-zero elements in each row of the adjacency matrix $A^k$. The input matrix $E_{u,k}^{k(0)} \in \mathbb{R}^{(M+N) \times 2dim}$ of the graph neural network at layer 0 is obtained by concatenating $E_{u}^k$ and $E_{u}^k$, which have personalized semantic information for behavior $k$.

The propagation rule for each layer of the graph neural network is:

$$E_{u,k}^{(l+1)} = \hat{A}^k \otimes E_{u,k}^{(l)}.$$  

(6)

Finally, the final graph neural network output embedding matrix is obtained as follows:

$$E_k = \text{mean}(E_{u,k}^{(0)}, E_{u,k}^{(1)}, ..., E_{u,k}^{(L)}).$$  

(7)

We split $E_k$ into two embedding matrices according to $[M, N]$, representing the final user embedding $E_u^k \in \mathbb{R}^{M \times 2dim}$ and item embedding $E_i^k \in \mathbb{R}^{N \times 2dim}$ respectively under behavior $k$:

$$E_u^k, E_i^k \leftarrow \text{split}(E_k, [M, N]).$$  

(8)

B. Behavior Relationship Denoising Module

The initial embeddings of each behavior cannot be uniformly scaled, which leads to different measurement scales of the high-order personalized semantics carried by each behavior type. In order to reduce the noise brought by multi-behavior interaction, we additionally treat all types of behaviors as a new behavior. To maintain the consistent shape with the high-order semantic output of each behavior, we concatenate the initial user embedding $E_u^k \in \mathbb{R}^{M \times \text{dim}}$ and item embeddings $E_i \in \mathbb{R}^{M \times \text{dim}}$ to themselves, resulting in new user embedding $E_u^k \in \mathbb{R}^{M \times 2dim}$ and new item embedding $E_i \in \mathbb{R}^{M \times 2dim}$:

$$\hat{E}_u = E_u \oplus E_u, \hat{E}_i = E_i \oplus E_i.$$  

(9)

After performing high-order connected semantic extraction on them, we obtain $E_u^k \in \mathbb{R}^{N \times 2dim}$ and $E_i^k \in \mathbb{R}^{N \times 2dim}$.

C. Dynamic Preferences Extraction Module

Most multi-behavior models that use graph neural networks consider only the interaction adjacency matrix between users and items, which ignores the varying user preferences for an item based on behavior frequency and order. To address this, we propose a dynamic preferences extraction module to capture the occurrence frequency of each behavior interaction and analyze dynamic preferences for optimizing prediction results.

Using a one-layer unidirectional GRU with ReLU activation function, we extract preferences information for each behavior interaction in each time period, analyze the changes in preference levels, and obtain $T_{u,i}^k \in \mathbb{R}^{d_{\text{ground}}}$, where $d_{\text{ground}}$ is the length of current time period output by this unidirectional GRU.

$$T_{u,i} = \text{ReLU} \left( \text{GRU} \left( T_{u,i} \right) \right).$$  

(10)

To obtain the initial preference coefficient $P_{u,i}^k$ for user $\bar{u}$ and item $\bar{i}$, we pass $T_{u,i}^k$ through a linear layer with input length $d_{\text{ground}}$ and output length 1.

To improve the robustness of the model, we optimize the initial preferences coefficients based on comparisons of $P_{u,i}$ for all items. The resulting final preferences coefficient matrix $P \in \mathbb{R}^{M \times N}$ represents user $\bar{u}$’s preferences for item $\bar{i}$, where $P_{u,i}$ is the coefficient. We first aggregate initial coefficients for all items using a Softmax layer to obtain $P_{u,i} \in \mathbb{R}^N$ for user $\bar{u}$.
diversity, we multiply $P_\hat{u}$ by $N$ to obtain the final preferences coefficient of user $\hat{u}$ for all items:

$$P_\hat{u} = \text{Softmax}(\hat{P}_u) \times N.$$ (11)

**D. Model Training**

Inspired by LightGCN [12], for model training using Mini-Batch method, the trainable parameters are divided into two parts: 1) the initial user and item embeddings $\Phi = \Phi^0$ that are affected by the Mini-Batch size, and 2) the other parameters $\Theta$. We use the Bayesian Personalized Ranking (BPR) loss, which is a pairwise loss that maximizes the difference between the positive samples (items with high user ratings) and negative samples (items with low user ratings) to improve the effectiveness of the recommender system. During the training phase, we optimize the following objective defined by using the Adam algorithm:

$$L_{BPR} = -\sum_{u \in U} \sum_{i \in N_u} \sum_{j \notin N_u} \ln \sigma(\hat{P}_{ui} - \hat{P}_{uj})$$

$$+ \lambda \left( \|\Phi^0\|^2 + \|\Theta\|^2 \right),$$ (12)

where $\lambda$ is the coefficient controlling L2 regularization and $\hat{P}_{ui}$ represents the final preferences prediction value of user $\hat{u}$ for item $i$. Its calculation formula is:

$$\hat{P}_{ui} = \tilde{v}_u \times I_{ui} = \sum_{k \in V} \tilde{e}_u^k \odot \tilde{e}_i^k + \sum_{k \in V} \tilde{e}_u^k \odot \tilde{e}_i^k \times P_{ui,k},$$ (13)

**V. Experiments**

This section conducts experiments on two real-world datasets to evaluate the performance of our MB-DP by comparing it with various recommendation techniques. Our goal is to answer the following questions:

- **RQ1**: How does MB-DP perform when competing with various recommendation baselines?
- **RQ2**: How do the sub-modules in MB-DP affect its recommendation performance?
- **RQ3**: How does different configurations of key hyperparameters affect the performance of MB-DP?

**A. Experimental Settings**

**Datasets.** We evaluate our model on two real-world datasets: UserBehavior Data from Taobao, an e-commerce platform in China, and IJCAI Data from Tmall, a shopping platform. UserBehavior Data includes four types of user-item interactions and corresponding timestamps, while IJCAI Data contains the shopping logs of anonymous users before and after "Double Eleven". We use the purchase behavior as the target behavior and summarized the dataset information in Table I.

**Evaluation Protocols.** For performance evaluation in top-N recommendation tasks, we use two widely adopted metrics: Normalized Discounted Cumulative Gain (NDCG@10) and Hit Rate (HR@10). We use leave-one-out evaluation for training and test set split, and reserve the first interaction after the last time period as the test dataset. We randomly select 99 uninteracted items for each user as negative examples.

**TABLE I. THE PREPROCESSED DATASETS**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>User#</th>
<th>Item#</th>
<th>Interaction#</th>
<th>Behavior types</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserBehavior</td>
<td>38042</td>
<td>28747</td>
<td>599977</td>
<td>{pv, fav, cart, buy}</td>
</tr>
<tr>
<td>IJCAI</td>
<td>11854</td>
<td>21356</td>
<td>184995</td>
<td>{pv, fav, cart, buy}</td>
</tr>
</tbody>
</table>

**Methods for Comparison.** To comprehensively evaluate the performance, we evaluate the performance of MB-DP by comparing it with various baselines from different research directions, including MF, NGCF [11], LightGCN [12], NGCF-M, LightGCN-M, and MB-GMN [6]. MF is a traditional matrix factorization approach. NGCF is a model that uses graph neural networks to generate node embeddings by propagating embeddings over the user-item bipartite graph, allowing for higher-order information aggregation. LightGCN is an improvement on GCN and is used in recommender systems to address training and prediction on large-scale graphs. NGCF-M and LightGCN-M treat all interaction behaviors as one and train using the NGCF and LightGCN models, respectively. MB-GMN is a model that uses graph meta-network technology to solve the problem of multi-behavior recommendation, learning the relationships between users and items using graph meta-networks and using attention mechanisms to handle relationships between different behaviors.

**Parameter Settings.** We use PyTorch to implement the MB-DP model, and optimize the parameters using the Adam optimizer with a learning rate of $1e^{-5}$. The regularization coefficient $\lambda$ is chosen from $\{0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$, and the batch size used during training is chosen from $\{32, 128, 256, 512, 1024, 2048\}$. By default, the hidden dimension of MB-DP is set to 64, the low-rank dimension is set to 4, and the hidden dimension of GRU is set to 16. The number of graph neural network layers in MB-DP is searched from $\{1, 2, 3, 4\}$. The neural network baselines used in the experiments are either implemented using their original code or with the above parameter settings.

**B. Performance Validation (RQ1)**

We evaluate the performance of predicting target user interaction behavior on different datasets and summarized the following observations:

Table II displays the performance of recommending top-N items for different target behavior types. From the results presented, we observe that MB-DP consistently achieves the best performance in top-N recommendations for different N values. This indicates that MB-DP has an advantage over other baselines in providing the correct interactive items and has a higher probability of doing so. Such performance differences can be attributed to its effective extraction of user multi-behavior information and dynamic preferences.

Our evaluation shows that incorporating user-item multi-behavior information into recommendation models improves performance, particularly with GNN-based models that capture higher-order relationships through stacked information propagation layers. We also find that MB-DP outperforms several multi-behavior recommendation model baselines, including MB-GMN and LightGCN, by effectively extracting dynamic preferences for multi-behavior. The significant
leads to consistently poor performance across various top-N patterns. 5) Dynamic preferences extraction each user and help in user behavior modeling. 3) Graph neural preferences. In contrast, the GNN-demonstrates the advantage of our method in capturing the performance gap between MB-DP and these baselines but ignore dynamic perception patterns or user behavior patterns but ignore dynamic preferences.

C. Model Ablation Study (RQ2)

To evaluate the design submodules of MB-DP, we consider five model variants: w/o MBeh, w/o Meta, w/o MGen, w/o Fuse, and w/o DI. These variants respectively remove the use of multiple behaviors in behavior semantic extraction, the low-order personalized semantic module, the high-order connected semantic extraction module, the behavior relationship denoising module, and the dynamic preferences extraction module.

Table III summarizes the results of ablation experiments on different datasets. The following conclusions can be drawn from the experimental results: 1) Using multi-behavioral interaction information is effective in this type of recommendation, as w/o MBeh performs poorly compared to MB-DP. 2) The performance of MB-DP is better than that of Meta, indicating that different behaviors have different meanings for each user and help in user behavior modeling. 3) Graph neural networks can differentiate between user-to-user or item-to-item behavior differences, showing the efficacy of incorporating interaction structure in behavior perception or user behavior patterns. 4) High-order connected semantic extraction by treating all behavior types as a new type reduces behavior noise, and the absence of behavior relationship denoising in w/o Fuse leads to consistently poor performance across various top-N recommendation settings. 5) Dynamic preferences extraction positively impacts recommendation performance as time

D. Hyperparameter Study (RQ3)

We experiment with different parameter settings for MB-DP to evaluate its performance with varying configurations of key hyperparameters: hidden state dimension, low-rank decomposition dimension, and number of layers in the graph neural network. The results are presented in Figure3-Figure5, where the y-axis shows the performance increase ratio compared to the default setting of MB-DP.

The hidden state dimension (dim) of MB-DP is varied from 8 to 64, and increasing dim improves the recommendation performance, but the improvement became relatively flat when dim > 32 for top-N recommendation beyond N = 1. Low-rank decomposition dimension (rank) is varied from 4 to 16, and the impact of rank on recommendation performance is found to be not significant. We also vary the number of graph neural network layers (layer) from 1 to 4 and find that increasing layers decrease the Top-1 recommendation performance due to overfitting, while increasing hit rate for Top-N recommendation with little effect on the recommendation order.
preferences contribute to the final recommendation performance. In future work, we will further explore how the connections between behaviors and the diverse semantics of behaviors affect the results show that MB-DP significantly outperforms various information about users’ dynamic interests for items. We conduct extensive experiments on two real-world datasets, and the results show that MB-DP significantly outperforms various state-of-the-art baselines. Furthermore, we find that dynamic preferences contribute to the final recommendation performance. In future work, we will further explore how the connections between behaviors and the diverse semantics of behaviors affect the recommendation quality, and seek general ways to mitigate this impact for better recommendation performance.

VI. CONCLUSION

In this paper, we propose a multi-behavior recommendation model integrating dynamic preferences (MB-DP), which reduces the impact of noise on recommendation performance when exploiting information from multiple types of interaction behavior and improves recommendation accuracy by utilizing information about users’ dynamic interests for items. We conduct extensive experiments on two real-world datasets, and the results show that MB-DP significantly outperforms various state-of-the-art baselines. Furthermore, we find that dynamic preferences contribute to the final recommendation performance. In future work, we will further explore how the connections between behaviors and the diverse semantics of behaviors affect the recommendation quality, and seek general ways to mitigate this impact for better recommendation performance.

REFERENCES


Session LMQA: Language Models and Question Answering
Abstract—The combination of pre-trained language models (LM) and knowledge graphs (KG) can enhance the reasoning ability for Question Answering. However, previous methods typically fuse the two modalities in a shallow or knowledge-draining manner, not taking full advantage of the knowledge representation of both. How to effectively fuse the different knowledge representations is still a problem of current research. In our work, a novel model is proposed that fuses LM modal knowledge representations and graph neural network (GNN) modal knowledge representations deeply over multiple layers of modality interaction operations. Specifically, the model includes an information interaction unit, through which KG and LM knowledge can be transferred between modalities to realize knowledge fusion directly, reducing information loss. In addition, we add the context node of implicit knowledge from LM encoding in the construction of the reasoning subgraph in advance for enhancing the reasoning of the GNN. We evaluate our model on two domains in the biomedical benchmark (MedQA-USMLE) and commonsense benchmarks (OpenBookQA and CommonsenseQA). Experimental results show that our model achieves a particular improvement over existing LM and LM+KG models for reasoning over both situational constraints and structured knowledge.

Keywords – Question Answering; GNN; LM; knowledge fusion.

I. INTRODUCTION

Question Answering is a challenging task for complex questions because it often contains multiple subjects, relations, and implicit background knowledge, and currently, the hot ChatGPT model is also conducting relevant research. Generally, knowledge can be encoded implicitly in a large unstructured pre-trained language model (LM) [1], or explicitly represented in structured knowledge graphs (KGs), such as ConceptNet [2]. Recently, the fine-tuning of large pre-trained language models generated by training on large text corpora on QA datasets has made great progress and has become the dominant paradigm for question-answering tasks [3], [4]. However, previous models are flawed in structured reasoning because they rely only on simple patterns (at times spurious) to reason about answers, rather than strong, structured reasoning that fuses explicit encyclopedic knowledge with implicit knowledge [5]. In other words, existing pre-trained language models for fine-tuning is the lack ability to exploit unambiguous encyclopedias and commonsense knowledge for reasoning [6].

Previous studies have shown that KGs are suitable for structured reasoning and play an essential role in structured reasoning (e.g., providing background knowledge) [7]–[9]. Therefore, extensive research exists to carefully design graph neural networks (GNN) to obtain answers by retrieving knowledge subgraphs, paths related to a given question by string matching, or semantic similarity or inference after modeling the retrieved subgraphs [10], [11]. However, using these inferences to retrieve graphs can introduce noise and limit the ability of models to effectively utilize both knowledge representations for reasoning [12].

Fig. 1: The left graph is based on context entities, while the right graph shows the result after inference. "Int" is a token used for knowledge exchange. Different node types are represented by colors that alternate during knowledge fusion. However, GNN may lose information during propagation, resulting in the inability to obtain distant information, as seen in the red box lacking Choice Entity knowledge.

To leverage the background knowledge provided by KGs to enhance LM representations, previous approaches combine these two models’ representations (i.e., expressive large language models and structured KGs) to improve inference performance [10], [13]. However, these methods typically fuse the two modalities in a shallow and non-interactive manner,
encoding both separately and combining them at the output for a prediction, or using one to augment the input of the other [11], [14], which limits the ability to exchange useful information between the two models. Latest research [14] attempted to fuse these two models’ representations by tokens, but it assumed that the GNN knowledge could be learned and aggregated to fixed token nodes, and ignored the problem of information loss during iterative message passing between neighbors on the graph (see Figure 1, the nodes enclosed by the red frame may lose the knowledge of choice entity nodes after reasoning). How to effectively fuse the representations from KG and LM remains an important open question.

To address the above issues, we propose a Language Models and Graph reasoning Fuse deeply for question answering model (LMGFuse) as shown in Figure 2, a new model which can deeply integrate and exchange the two model representations in multi-layer architecture for multimodal fusion. Our proposed LMGFuse is mainly composed of two parts: one is the pre-trained language model to encode and understand the QA context; the other is the modal interaction of attention-based GNN and LM for joint reasoning. The former is to generate implicit background knowledge of the context, and the latter is to fuse implicit knowledge and explicit knowledge for reasoning. Referring to previous studies [11], we use the LM to encode the QA context to generate implicit knowledge, and combine this knowledge with the QA context entities to retrieve a KG subgraph following prior works [13]. After that, the LM knowledge representation and subgraph are fed into the model fusion layer, which will fuse the token node information output by the LM encoder with all node information in GNN. Through these layers, each node of the subgraph can learn the knowledge from LM directly, and the LM encoder can also learn the subgraph knowledge, reducing knowledge loss. Meanwhile, to reduce the number of parameters in modal fusion, dimension reduction of parameters is carried out by factoring, improving also the efficiency of the operation.

The contributions of this paper are three-fold: (1) an innovative approach to achieve knowledge representation fusion between LM and GNN. (2) innovative use of reasoning subgraph construction and parameter reduction techniques in question answering. (3) the experimental results on two domains with three datasets (CommonsenseQA [15], OpenBookQA [16] and MedQA-USMLE [17]) are better than the existing LM and LM+KG fusion models.

II. RELATED WORK

There are two main research methods for QA systems under complex problems in prior work: semantic parsing-based (SP-based) and information retrieval-based (IR-based) [18], [19]. SP-based reasoning methods based on the fusion of LM and GNN knowledge representations have made great progress and become a hot research topic. Some works use two-tower to fuse the representations, but they lack contralateral information interaction or have information loss [20]. Others attempt to use one pattern to enhance the other serially, such as using the last layer of the LM knowledge representation to enhance the GNN structured representation or using the GNN structured representation to heighten the context representation [10], [13], [21]. However, in previous works, the interaction mode between the two models was limited, because the information between them could not be interacted and fused, but only flowed in one direction [14].

Several studies aim to fuse information from two models at a deeper level. Some of these works [22] use LM implicit knowledge combined with GNN model structured reasoning to construct QA data for inferencing. However, these methods focus too much on the reasoning of implicit knowledge. Recently, GREASELM [14] and QA-GNN [11] proposed updating the LM representations and GNN representations jointly through message passing. Nevertheless, their method of jointly updating knowledge representations does not handle it well: QA-GNN uses single-pool representations without deep fusion, and GREASELM updates the representation with information loss. In our work, we keep the token node representation of the LM and the graph node representations for deep fusion and use this token node to exchange information with each node in the graph reducing information loss.

In addition, some studies have explored ways to enhance the representation of LM with explicit knowledge from KG in the pre-training stage. However, this modal interaction is limited to the knowledge representation provided [23] and does not fully utilize the structured reasoning ability of the two models.

III. APPROACH: LMGRFQA

As shown in Figure 2, the input to the model is the QA context [q; a] concatenated between question q and candidate answer a. LMGRFQA works as follows. First, we use LM encode representations of QA context as context nodes and retrieve KG based on QA context entities to build subgraphs containing implicit knowledge. Then, before modal fusion, we use an N-layer LM encoder to the QA context for easier knowledge modal transfer through token nodes later in the first modal interaction layer. In modal knowledge fusion, we maintain the independent structures of LM and GNN and use the designed Exchange GNN and LM’s Int token representation unit (EXGLInt) to cross-fusion after each layer to update the knowledge representation obtained by each model. After multi-layer interaction, each node representation can learn the knowledge representation of two modalities. Finally, we make a final prediction using the LM token node representation and GNN node representations through the pooling and MLP layers.

A. Subgraph Construction

In the process of subgraph retrieval and construction, given a [q; a], we retrieve a KG subgraph following prior works [13], [14] and follow the settings of [11] to divide subgraph nodes into four categories: question nodes, option nodes, context nodes, and other knowledge nodes, respectively (corresponding to the node color, green, purple, pink and blue in the subgraph of Figure 2) to capture the strength of association between two nodes. We set up implicit knowledge of the QA
context as the subgraph one node by pooling and joint in advance for enhancing the reasoning of GNN, where we use a special LM encoder for enhancing the representation such as Roberta-Large [24] for CommonsenseQA, AristoRoBERTa [25] for the OpenbookQA and so on. The node connects the QA context node to each question entity and answers entity nodes.

B. Language Pre-Representation

For the sequence of QA context embeddings \( \{w_{int}, w_1, ..., w_T\} \), after fed through a special LM encoding layer, we use an N-layer LM encoder to encode it into a language representation \( \{h_{int}^n, h_1^n, ..., h_T^n\} \), incorporating location information, etc. We opt to use the BERT [26] layer as the N-layer LM encoder due to its relatively smaller parameter size compared to other LM encoders.

\[
\{h_{int}^i, h_1^i, ..., h_T^i\} = LM(\{h_{int}^{i-1}, h_1^{i-1}, ..., h_T^{i-1}\}) \quad (1)
\]

for \( i = 1, ..., N \)

where \( LM(\cdot) \) is a single-layer pre-trained language model with parameter initialization loaded in advance, \( h_{int}^i \) represents the encoded knowledge representation of the token node at layer \( i \), which interacts with the GNN node representation to exchange information. More technical implementation details need to refer to [11].

C. Graph Inference Representation

We take the embedding \( \{e_1^0, ..., e_T^0\} \) of the constructed subgraph node as input which is initialized from the LM and construct the subgraph by referring to the construction method of the Graph Attention Framework (GAT) [27]. The node representation after \( l \) layer is calculated by the following formula.

\[
\{e_1^l, ..., e_{T-1}^l, e_T^l\} = GNN(\{e_1^{l-1}, ..., e_{T-1}^{l-1}, e_T^{l-1}\}) \quad (2)
\]

for \( l = 1, ..., M \)

where \( GNN(\cdot) \) represents GAT, and its technical design scheme follows from [11], [14]. GNN calculates and updates the knowledge representation of each node \( e_j \in \{e_1, ..., e_J\} \) through its neighbor nodes knowledge representation, node’s type, and edge information.

\[
e_j^l = f_n(\sum_{i \in N_j \cup \{e_j\}} \alpha_{ij}m_{ij}) + e_j^{l-1} \quad (3)
\]

where \( N_j \) represents the set of \( e_j \) neighbor nodes, \( \alpha_{ij} \) denotes the attention weight of message passing, \( m_{ij} \) represents the message from neighbor node \( i \) to node \( j \), and \( f_n \) represents the multi-layer MLP. Information about the neighbor node of node \( j \) such as the relation type and node representation are aggregated to \( m_{ij} \), and it is calculated by the following formula.

\[
r_{ij} = f_r(\tilde{r}_{ij}, u_i, u_j) \quad m_{ij} = f_m(e_i, u_i, r_{ij}) \quad (4)
\]
where $r_{ij}$ is relation embedding from node $i$ to node $j$, $u_i$ and $u_j$ are the node type embedding of nodes $i$ and $j$, $\tilde{r}_{ij}$ is a relation embedding for the relation connecting $e_i$ and $e_j$, $f_r$ is a multi-layer MLP, and $f_m$ is a linear transformation. $\alpha_{ij}$ reflects the importance of the neighbor node $i$ to the message of node $j$, and its calculation formula is as follows.

$$q_i = f_q(e_i, u_i) \quad k_j = f_k(e_j, u_j, r_{sj})$$

$$\gamma_{ij} = \frac{q_i^T k_j}{\sqrt{D}} \quad \alpha_{ij} = \frac{\exp(\gamma_{ij})}{\sum_{e_i, e_N, j \in N_j} \exp(\gamma_{ij})}$$

where $u_i$, $u_j$, $r_{sj}$ are defined as above, $D$ is graph node encode dimensions, and $f_q$, $f_k$ are linear transformation. More technical implementation details need to refer to the GAT [27].

### D. Modal Knowledge Interaction

After two independent knowledge representation layers of LM and GNN, we use an EXGLInt for modal interaction, which combines the token representation of LM with the representation of each GNN node.

$$[h_{int}^i; e_i^j] = EXGLInt([h_{int}^i; e_i^j])$$

where $T$ represents the number of subgraph nodes and $h_{int}^i$, $e_i^j$ are defined as above. In EXGLInt, we use multiple layers of MLP as information exchange units and use a two-layer pooling layer to degrade an excessive number of parameters in combination with factorization ideas [28]. LM knowledge representation does not participate in the interaction of GNN representation except with the token node directly, and the token node conducts knowledge interaction with each node representation of GNN. Through modal interaction, each node of GNN can learn LM modal knowledge, and LM can also learn GNN modal knowledge from multi-layer interaction (see Figure 2, the color fusion of node).

### E. Reasoning and Prediction

After modal fusion of LM and GNN, the obtained $h_{int}^M$ knowledge representation and graph node knowledge representation $\{e_1^M, ..., e_T^M, e_T^F\}$ are concatenated through the pooling layer. Then the representation is fed into MLP and softmax to score a given (question, answer choice) pair based.

$$p = (a|q) = MLP(h_{int}^M, e_1^M, ..., e_T^M, e_T^F)$$

We use the cross-entropy loss and RAdam optimizer to optimize the whole model end-to-end.

### IV. Experiments Settings

Following previous work [11], [14], we set the batch size to 128 and use mini-batch training. We set separate learning rates for GNN and LM, where the learning rate for GNN is chosen from $\{1 \times 10^{-3}, 2 \times 10^{-3}\}$ and the learning rate for LM is chosen from $\{1 \times 10^{-5}, 3 \times 10^{-5}\}$ [12]. The pre-trained language model uses parameters provided by the Pytorch interface in advance for parameter initialization. Given each query, we set the number of subgraph retrieval hops to 2 according to [13] and the number of nodes reserved for each subgraph to 200 following previous work [11], where we set the node dimension of the graph to 200 and number of layers (N = 5) of our GNN module [14]. We use one GPU (GeForce RTX 3090 Ti-24g) for our experiments, and each task takes about 10 hours.

#### A. DataSets

We evaluate our model LMGFuse on three standard QA benchmarks: CommonsenseQA [15], OpenBookQA [16], and MedQA-USMLE [17], which come from different domains (commonsense and medical).

**CommonsenseQA** is a 5-option commonsense question answering dataset of 12,102 questions that requires commonsense knowledge for reasoning. We conduct experiments on the in-house data split of [10] to compare to baseline methods since the CommonsenseQA test set is not publicly available.

**OpenbookQA** is a 4-option question commonsense answering dataset of 5957 questions that require scientific facts knowledge for reasoning. We use the official data-splitting method [16].

**MedQA-USMLE** is a 4-option medical question answering dataset of 12,723 questions that require biomedical and clinical knowledge for reasoning. The data segmentation method refers to the official paper [17].

#### B. Knowledge Graph

We use ConceptNet [2] as the knowledge source for CommonsenseQA and OpenBookQA which is a general-domain knowledge graph and better suited to commonsense reasoning tasks. It has 799,273 nodes and 2,487,810 edges in total. For MedQA-USMLE, We use the knowledge graph constructed by [11], [14]. It contains 9,958 nodes and 44,561 edges.

#### C. Language Models

We set up different language models for different domain tasks to better reason the implicit knowledge of each domain task. We use the Roberta-Large [24] in CommonsenseQA, and AristoRoBERTa [25] in OpenbookQA, which are commonsense pre-trained language models. We use the SapBERT [29] on MedQA-USMLE, which is a biomedical pre-trained language model. These language models selected demonstrate LMGFuses generality concerning for to language model initializations.

### V. RESULTS AND ANALYSIS

#### A. Main Results

Our experimental results on CommonsenseQA and OpenBookQA datasets are presented in Table 1, respectively. From the table comparison results of the previous two datasets, our model has improved performance compared to the fine-tuned LM and the existing LM+KG model, on CommonsenseQA, compared to Roberta +5.7% and the previous best LM+KG model GREASELM compared to +0.2%, and on OpenBookQA, obvious with 7.6% higher than fine-tuned LMs.
and 1.2% higher than LM+KG models. Improvements over QA-GNN and GREASELM show that our model LMGFuse outperforms the LM+KG approach in transferring information between text and KG representations. QA-GNN does not integrate continuous interactions between two modalities, and GREASELM uses more expressive labeled interactions to fuse interactions that limit the ability of GNNs to propagate information. The results on the OpenbookQA leaderboard are shown in Table 2. UnifiedQA (11B params) and T5 (3B) are about 30x and 8x larger than our model.

TABLE I: Evaluation of our models on CommonsenseQA and OpenBookQA datasets under the same random seed. For CommonsenseQA, as the official test is hidden, so here we report the in-house Test (IHiTest) accuracy and use the same data set as [11], [14].

<table>
<thead>
<tr>
<th>Methods</th>
<th>CommonsenseQA Acc.(%)</th>
<th>OpenBookQA Acc.(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa-Large (w/o KG)</td>
<td>68.7</td>
<td>78.4</td>
</tr>
<tr>
<td>RoBERTa (no KG)</td>
<td>-</td>
<td>78.4</td>
</tr>
<tr>
<td>RGCN [30]</td>
<td>68.4</td>
<td>74.6</td>
</tr>
<tr>
<td>GconAttn [20]</td>
<td>68.6</td>
<td>71.8</td>
</tr>
<tr>
<td>MHGRN [13]</td>
<td>71.1</td>
<td>80.6</td>
</tr>
<tr>
<td>QA-GNN [11]</td>
<td>73.4</td>
<td>82.8</td>
</tr>
<tr>
<td>GREASELM [14]</td>
<td>74.2</td>
<td>84.8</td>
</tr>
<tr>
<td>LMGFuse (Ours)</td>
<td><strong>74.4</strong></td>
<td><strong>86.0</strong></td>
</tr>
</tbody>
</table>

In addition, results on public datasets show that our model exhibits superior performance in modal fusion, for which we study performance on MedQA-USMLE datasets from other domains. Table 3 shows that our model also has a better performance compared to classical LM methods (e.g. SapBERT [29]) in the biomedical domain compared with QA-GNN and GREASELM, with a 4.3% improvement over fine-tuned GNN and a 2.5% improvement over the LM+KG model.

Stacked of Modal Fusion Layer We test the effect of the number of modal interaction layers on model performance. As shown in Figure 3, increasing the number of modal interaction layers continues to bring benefits until the number of layers \( N = 3 \), when \( N > 3 \), the performance begins to degrade. As the number of layers increases, the model changes from underfitting to overfitting.

TABLE II: Test accuracy on OpenBookQA leaderboard

<table>
<thead>
<tr>
<th>Methods</th>
<th>Acc.(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AristoRoBERTaV7</td>
<td>77.8</td>
</tr>
<tr>
<td>QA-GNN-DeBERTa</td>
<td>79.0</td>
</tr>
<tr>
<td>QA-GNN [11]</td>
<td>82.8</td>
</tr>
<tr>
<td>GREASELM [14]</td>
<td>84.8</td>
</tr>
<tr>
<td>T5 11B + KB</td>
<td>85.4</td>
</tr>
<tr>
<td>JointLK [12]</td>
<td>85.6</td>
</tr>
<tr>
<td>UnifiedQA (11B) [4]</td>
<td>87.2</td>
</tr>
<tr>
<td>LMGFuse (Ours)</td>
<td><strong>86.0</strong></td>
</tr>
</tbody>
</table>

TABLE III: Performance on MedQA-USMLE.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Acc.(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIOROBERTA-BASE</td>
<td>36.1</td>
</tr>
<tr>
<td>BIOBERT-LARGE</td>
<td>36.7</td>
</tr>
<tr>
<td>SapBERT-Base (w/o KG)</td>
<td>37.2</td>
</tr>
<tr>
<td>QA-GNN</td>
<td>38.0</td>
</tr>
<tr>
<td>GREASELM</td>
<td>38.5</td>
</tr>
<tr>
<td>LMGFuse (Ours)</td>
<td><strong>41.0</strong></td>
</tr>
</tbody>
</table>

unfreeze epoch We researched the unfreeze epoch, a hyperparameter that affects the parameter updates of the LM model during backpropagation. We found that freezing the LM parameters for a certain number of epochs can improve the performance of the model. As shown in Figure 4, on OpenBookQA, the performance of the model increases by about 2% when the unfreeze epoch is set from 0 to 4, and when it is set from 4 to 6, it decreases by about 1.6%.

Furthermore, we do not compare with models on higher leaderboards on OpenBookQA, such as unified QA [4], and Albert+DESC-KCR [31], because they either use stronger text encoders or use additional data resources, while our model focuses on improving joint reasoning between KG and LM.

B. Ablation studies

Through ablation experiments, we analyze the effectiveness of different model components on the MedQA-USMLE dataset, which includes rich background information. We evaluate the effect of fusing retrieval subgraphs with QA contextual information on model performance.

TABLE IV: The performance of LMGFuse with and without modal fusion on MedQA-USMLE.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Acc.(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREASELM (No QA context)</td>
<td>38.5</td>
</tr>
<tr>
<td>GREASELM (Join QA context)</td>
<td>38.8</td>
</tr>
<tr>
<td>LMGFuse (No QA context)</td>
<td>39.3</td>
</tr>
<tr>
<td>LMGFuse (Join QA context)</td>
<td><strong>41.0</strong></td>
</tr>
</tbody>
</table>

Graph Construction with QA context We perform experiments to test whether adding QA context nodes to the model can improve the performance of the model under the condition that other environments such as random seeds are consistent. We do not consider splicing the QA context node with all
nodes at the beginning, because the subsequent modal fusion process is similar to this operation. The results in Table 4 show that graph reasoning with QA context nodes can bring certain improvements.

VI. CONCLUSION

In this paper, we propose the LMGFuse model, a new model that realizes the multi-level deep interaction and fusion of LM knowledge representations and GNN knowledge representations in a novel way. In this model, we design a deep interaction and fusion module, so that information can be transferred and updated between the two knowledge models. In addition, we also added the context node of the implicit knowledge generated from LM encoding in the construction of the reasoning subgraph in advance, so that the GNN can learn the implicit knowledge during the first message-passing process and enhance the reasoning ability of the GNN. We conduct experiments on multiple domains (commonsense and medical) datasets, and the results show that our model outperforms the previous KG+LM and LM-only baselines, demonstrating the models’ generality with respect to language model initializations.

ACKNOWLEDGEMENTS

This work was in part supported by NSFC (Grant No.62176194), the Science and Technology Innovation 2030 (Grant No. 2022ZO12060604) Sanya Science and Education Innovation Park of Wuhan University of Technology (Grant No.2020KF0057) and Fundamental Research Funds for the Central Universities (WUT: 2021III054JC).

REFERENCES

KLAPrompt: Infusing Semantic Knowledge into Pre-trained Language Models by Long-answer Prompt Learning

Zuotong Xie¹, Kai Ouyang¹, Xiangjin Xie¹, Hai-Tao Zheng¹,³*, Wenqiang Liu², Dongxiao Huang², Bei Wu²
¹Shenzhen International Graduate School, Tsinghua University, China
²Interactive Entertainment Group Tencent Inc, China
³Pengcheng Laboratory, Shenzhen, China

Abstract—Pre-trained language models (PLMs) with external knowledge have demonstrated their remarkable performance on a variety of downstream natural language processing tasks. The typical methods of integrating knowledge into PLMs are designing different pre-training tasks and training from scratch, which requires high-end hardware, massive storage resources, and computing time. Prompt learning is an effective approach to tune PLMs for specific tasks, and it can also be used to infuse knowledge. However, most prompt learning methods accept one token as the answer instead of multiple tokens. To tackle this problem, we propose the long-answer prompt learning method (KLAPrompt) to incorporate semantic knowledge in Xinhua Dictionary into pre-trained language models. The proposed method splits the whole answer space into several answer subspaces according to the token’s position in the long answer. Extensive experimental results on five datasets demonstrate the effectiveness of our approach.

Index Terms—semantic knowledge, pre-trained language model, prompt learning

I. INTRODUCTION

In recent years, pre-trained language models (PLMs) with external semantic knowledge have shown excellent performance on many natural language processing (NLP) tasks, including named entity recognition [1]–[4], relation extraction [5]–[8], and machine translation [9]–[12]. However, traditional approaches of introducing knowledge are mostly training from scratch, which is time-consuming and computationally expensive, making it infeasible for most users. Recently, prompt learning has achieved promising results for certain few-shot classification tasks [13]–[16], and it can also be used to integrate knowledge. Xinhua Dictionary [17], the most authoritative and influential modern Chinese dictionary, contains massive and comprehensive content such as wordforms, pronunciations, precise definitions, and rich examples. As shown in Table I, the “sense” is composed of a long string of tokens, but the typical methods of prompt learning accept one token as the answer.

To address this challenge, we propose the long-answer prompt learning method (KLAPrompt) and collect a word sense prediction dataset (WSP) based on Xinhua Dictionary to introduce fine-grained semantic knowledge. Firstly, instead of considering the long answer as a whole, we split the answer space into several answer subspaces according to the token’s position in the long answer. For instance, the answer subspaces of “order” in Table I are {“the”}, {“way”}, {“state”, “request”}, {“in”, “that”, “for”}, . . . , {“for”}. Then, we train pre-trained language models on WSP dataset to predict the sense, and each word of the sense will be predicted independently.

We conduct comprehensive experiments on five public NLP datasets. Experimental results demonstrate that pre-trained language models gain superior performances on the strength of the semantic knowledge in Xinhua Dictionary. And empirical studies also verify the effectiveness of the KLAPrompt approach in integrating semantic knowledge.

In a nutshell, the main contributions of our work are as follows:

1) We introduce more abundant and fine-grained semantic knowledge in Xinhua Dictionary into the pre-trained language models, enhancing the model’s ability to understand Chinese word semantics.

<table>
<thead>
<tr>
<th>Word</th>
<th>Sense</th>
<th>Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>the state that exists when people obey laws, rules or authority</td>
<td>keep the class in good order</td>
</tr>
<tr>
<td></td>
<td>a request for food or drinks in a restaurant; the food or drinks that you ask for</td>
<td>May I take your order?</td>
</tr>
</tbody>
</table>

TABLE I: An example of the word, senses, and phrases in the dictionary.
2) We propose a novel long-answer prompt learning method (KLAPrompt), which provide a reasonable solution for two main challenges for answer engineering: (a) When there are many classes, how to seek the proper answer space? (b) How to decode the multi-token answers?

3) Extensive experiments on five Chinese NLP tasks demonstrate the proposed method significantly empowers the widely-adopted pre-trained language models. The empirical studies also confirm that the KLAPrompt with "sense" knowledge gains more significant improvement with less fine-tuning data.

4) We collect a word sense prediction dataset (WSP) based on Xinhua Dictionary, which is available at https://github.com/Xie-Zuotong/WSP

II. RELATED WORK

A. Semantic Knowledge

Semantic knowledge contains the meaning of words, phrases, and sentences, examining how meaning is encoded in a language. It has been extensively used in various natural language processing tasks [18]–[21].

ERNIE [22] has improved BERT’s masking strategy to integrate entity information in the knowledge graph. In Chinese, an entity or phrase is composed of several Chinese words. If only a single word is masked, the model can easily predict the masked content only through the context information, without paying attention to the composition of phrases and entities, as well as the syntactic and semantic information in sentences. Therefore, ERNIE masks all tokens that compose a whole phrase or entity at the same time. However, the phrase in ERNIE usually consists of two or three tokens. When the number of consecutive tokens exceeds twenty, the model is difficult to train, and the performance will decline. KnowBERT [23] integrates WordNet [24] and a subset of Wikipedia into BERT and uses the Knowledge Attention and Recontextualization mechanism to explicitly model entity spans in the input text. SenseBERT [25] adds a masked-word sense prediction task as an additional task to learn the "sense" knowledge in WordNet. WordNet lexicographers organize all word senses into 45 supersense categories. Hence, it predicts not only the masked words but also their supersenses during pre-training. Both KnowBERT and SenseBERT introduce WordNet into the BERT, but compared with Xinhua Dictionary or Oxford Dictionary, the supersenses in WordNet are relatively limited, and the word meaning is coarse-grained.

Furthermore, most of these methods are training from scratch, which is time-consuming and computationally expensive, making it infeasible for most users.

B. Prompt Learning

Prompt learning is based on the language model used to calculate the probability of text [26]. Unlike adapting pre-trained language models to downstream tasks through objective engineering, prompt learning utilizes additional textual prompts to make downstream tasks look more like those solved during the original language model training.

Radford et al. [27] illustrate that language model can learn NLP tasks without direct supervision, and then prompt learning has gradually become the most popular research direction in natural language processing. Prompt learning includes prompt engineering and answer engineering. For discrete prompts, Brown et al. [28] manually create prefix prompts to deal with diverse natural language processing tasks. For continuous prompts, P-tuning [13] proposes prompts learned by inserting trainable variables into the embedded
input. Recent work [14] manually designed the constrained answer spaces for Named Entity Recognition tasks.

But there are still two challenges for answer engineering: (a) When there are many classes, how to seek the proper answer space? (b) How to decode the multi-token answers?

III. METHODOLOGY

In this section, we introduce the KLAPrompt approach and its detailed implementation. There are two steps in our KLAPrompt method: prompt engineering and answer engineering. So we elaborate our method from these two aspects.

A. Prompt Engineering

Prompt engineering, also known as template engineering, is to design a prompting function that results in the most effective performance on the downstream task. It is based on the language model used to calculate the probability of text [29], and it utilizes additional prompts to make downstream tasks look more like those solved during the original language model training.

A template is a textual string with two slots: an input slot [X] for input x and an answer slot [Y]. For example, in the case of sentiment analysis where $x = \text{“I love this movie.”}$, the template may take a form such as “[X] Overall, it was a [Y] movie.”. Then, the prompt would become “I love this movie. Overall, it was a [Y] movie.”. The number of [X] and [Y] slots can be flexibly changed for the need of tasks at hand.

In many cases, these template words are not necessarily composed of natural language tokens; they could be virtual tokens that would be embedded in a continuous space later and optimized through gradient descent.

In our work, we use some auxiliary virtual tokens $[P_1], [P_2], \ldots, [P_l]$, whose parameters are randomly initialized, to make the template more effective, and $l$ is a predefined hyper-parameter. This method performs prompting directly in the embedding space of the model.

The word sense prediction dataset (WSP) contains the word [W], sense $y$, and sentence $x$ for each example. For the continuous prompt in WSP dataset, we first copy the word [W] mentioned in the sentence $x$, then add a few auxiliary virtual tokens followed by the answer slot [Y] that the model will predict and the input slot [X]. There is an example of the continuous prompt in Figure 1, and the complete prompt becomes:

$$T(x) = \[W][P_1], [P_2], \ldots, [P_l][Y],[X]$$ (1)

where $T(\cdot)$ is the template for WSP dataset, [W] is the word mentioned in the input sentence $x$, $[P_i]$ is the virtual token, $[X]$ is the input slot for sentence $x$, and $[Y]$ is the answer slot for sense $y$. Each embedding of prompts is randomly initialized and optimized during training.

B. Answer Engineering

Unlike prompt engineering, which discovers suitable prompts, answer engineering tries to seek a proper answer space and a map to the original output that brings about an effectual predictive model. For classification-based tasks, there are two main challenges for answer engineering: (a) When there are too many classes, how to select an appropriate answer space becomes a difficult combinatorial optimization problem. (b) When using multi-token answers, how to best decode multiple tokens using PLMs remains unknown [26]. In this section, we propose the long-answer strategy to address the challenges mentioned above.

In prompt learning, for each class $y \in \mathcal{Y}$, the mapping function $\phi(\cdot)$ will map it to the answer $\phi(y) \in \mathcal{Y}$, where $\mathcal{Y}$ is the answer space. It’s easy to find the appropriate answer space and the mapping function when the classes are limited, and all the answer consists of a single token. Unfortunately, there are massive classes in WSP dataset (It includes 7,390 words and 16,495 senses; each word has one to thirteen senses), and the answer is quite long sometimes. Take the word “order” as an example. The template and the label word set can be formalized as:

$$\{\text{“the way in which people or …”}, \text{“the state that exists when …”}, \text{“a request for food or drinks …”}\}$$

But the pre-trained language model like BERT [29] cannot predict the whole long answer at once. So in our work, we split the answer space $\mathcal{Y}_{[\text{MASK}]}$ into several answer subspaces $\{\mathcal{Y}_{[\text{MASK}]}_1, \mathcal{Y}_{[\text{MASK}]}_2, \ldots, \mathcal{Y}_{[\text{MASK}]}_j, \ldots, \mathcal{Y}_{[\text{MASK}]}_n\}$ according to the token’s position in the answer, where $n$ is the length of the answer, and $\phi_j(y)$ is to map the class $y$ to the set of label words $\mathcal{Y}_{[\text{MASK}]}_j$ for the $j$-th masked position $[\text{MASK}])_j$. Here we still take the word “order” as an example. As shown in Figure 1, the template and the label word set can be formalized as:

$$T(x) = \[W][P_1], \ldots, [P_l][\text{MASK}].x\$$

$$\mathcal{Y}_{[\text{MASK}]}_1 = \{\text{“the”, “a”}\}$$

$$\mathcal{Y}_{[\text{MASK}]}_2 = \{\text{“way”, “state”, “request”}\}$$

$$\mathcal{Y}_{[\text{MASK}]}_3 = \{\text{“in”, “that”, “for”}\}$$

$$\mathcal{Y}_{[\text{MASK}]}_4 = \{\text{“which”, “exists”, “food”}\}$$

$$\ldots$$

In a conventional supervised learning system for natural language processing, we take an input $x \in \mathcal{X}$ and predict an output $y \in \mathcal{Y}$ based on the language model $p(y|x)$. As the template may contain multiple [MASK] tokens, we must consider all masked positions to make predictions, i.e.,

$$p(y|x) = \prod_{j=1}^{n} p([\text{MASK}]_j = \phi_j(y)|T(x))$$ (4)

where $n$ is the number of masked positions in $T(x)$, and $\phi_j(y)$ is to map the class $y$ to the set of label words $\mathcal{Y}_{[\text{MASK}]}_j$ for the $j$-th masked position $[\text{MASK}])_j$. Equation 4 can be used to tune PLMs and classify classes.
TABLE II: Experiment results of baselines and our methods on five datasets (Acc.%). “+ KLAPrompt” means that we train PLMs with KLAPrompt method via semantic knowledge infusion training before fine-tuning.

<table>
<thead>
<tr>
<th>Models</th>
<th>STS-B</th>
<th>Book Review</th>
<th>XNLI</th>
<th>Chnsenticorp</th>
<th>IFLYTEK</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>50.75</td>
<td>86.62</td>
<td>76.8</td>
<td>93.3</td>
<td>60.52</td>
</tr>
<tr>
<td>BERT + KLAPrompt</td>
<td>52.92</td>
<td>88.63</td>
<td>78.61</td>
<td>94.82</td>
<td>61.58</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>48.23</td>
<td>89.08</td>
<td>78.37</td>
<td>94.85</td>
<td>60.44</td>
</tr>
<tr>
<td>RoBERTa + KLAPrompt</td>
<td>50.37</td>
<td><strong>91.12</strong></td>
<td>80.69</td>
<td>95.1</td>
<td>61.46</td>
</tr>
<tr>
<td>MacBERT</td>
<td>52.92</td>
<td>88.78</td>
<td>79.05</td>
<td>94.98</td>
<td>60.82</td>
</tr>
<tr>
<td>MacBERT + KLAPrompt</td>
<td><strong>54.67</strong></td>
<td><strong>90.1</strong></td>
<td><strong>81.49</strong></td>
<td><strong>95.79</strong></td>
<td><strong>62.01</strong></td>
</tr>
</tbody>
</table>

With the pre-trained language model predicting the masked tokens, the loss function of KLAPrompt is given by:

\[
\mathcal{L} = -\frac{1}{|X|} \sum_{x \in X} \log p(y|x) = -\frac{1}{|X|} \sum_{x \in X} \log \prod_{j=1}^{n} p([\text{MASK}]_j = \phi_j(y)|T(x))
\]

IV. EXPERIMENTS

In this section, we present the details of implementation and conduct experiments on five Chinese NLP datasets to evaluate the efficiency and effectiveness of our approach.

A. Datasets

**STS-B.** This Chinese version of the dataset \(^1\) is translated from the original English dataset STS-B [30] and partially manually revised. Semantic Textual Similarity (STS) measures the meaning similarity of sentences.

**Book Review.** Book Review dataset [31] is collected from Douban, a Chinese online review website that provides information about books, movies, and music. It’s a one-sentence text classification dataset.

**XNLI.** In our experiment, only the Chinese part of the Cross-language Natural Language Inference dataset (XNLI) [32] is retained. In XNLI, the model should read the two sentences and determine whether the relationship between them is “Entailment”, “Contradiction”, or “Neutral”.

**Chnsenticorp.** Chnsenticorp [31] is a sentiment analysis dataset which contains 12,000 hotel reviews. 6,000 reviews are positive, and the other 6,000 reviews are negative.

**IFLYTEK.** The IFLYTEK [33] dataset has more than 17,000 long texts about the application description, including various application topics related to daily life with a total of 119 categories.

Datasets above are with 8.05K, 40.0K, 40.0K, 12.0K, and 17.3K samples respectively. We follow the evaluation metrics and setting used in [31, 33].

---

\(^1\)https://github.com/pluto-junzeng/CNSD

---

TABLE III: Ablation study on XNLI dataset (Acc.%). “+ WSP” means that we train BERT on WSP dataset without the KLAPrompt approach. † means that we train these models on WSP dataset before fine-tuning.

<table>
<thead>
<tr>
<th>Models</th>
<th>XNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>76.8</td>
</tr>
<tr>
<td>- BERT + WSP(^1)</td>
<td>77.34 (+0.54)</td>
</tr>
<tr>
<td>- BERT + Continuous Prompt(^1)</td>
<td>77.72 (+0.92)</td>
</tr>
<tr>
<td>- BERT + Long-answer Strategy(^1)</td>
<td>78.17 (+1.37)</td>
</tr>
<tr>
<td>BERT + KLAPrompt(^1)</td>
<td><strong>78.61</strong> (+1.81)</td>
</tr>
</tbody>
</table>

B. Implementation Details

KLAPrompt is based on pre-trained language models. In this work, we choose BERT [29], RoBERTa [34], and MacBERT [35] as our basic models. For all these models, the number of layers is 12, the hidden size is 768, the number of heads is 12, and it contains 110M parameters. These models are optimized by Adam optimizer [36] with the initial learning rate of 1e-5. The training batch size is 64. Each model is trained for 10 epochs and evaluated on the validation set for every epoch. All experiments are carried out using a single NVIDIA GeForce RTX 3090 24GB card.

C. Main Results

The experimental results on the development set of five Chinese natural language processing datasets are presented in Table II. We show each original model and the model trained with KLAPrompt method (e.g., BERT and BERT + KLAPrompt). We find that all pre-trained language models trained with KLAPrompt method have achieved significant improvement compared to the original PLMs. For STS-B, Book Review, and XNLI datasets, RoBERTa + KLAPrompt pushes up the final results by 2.14%, 2.04%, and 2.32%. And for IFLYTEK dataset, the method still can raise the accuracy by more than 1%. This superior performance proves that infusing
TABLE IV: Model performance on Chnsenticorp dataset (Acc.%) w.r.t. different values of hyper-parameter $l$.

<table>
<thead>
<tr>
<th>Setting</th>
<th>$l = 1$</th>
<th>$l = 2$</th>
<th>$l = 3$</th>
<th>$l = 4$</th>
<th>$l = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT + KLAPrompt</td>
<td>94.24</td>
<td>94.82</td>
<td>94.65</td>
<td>94.57</td>
<td>94.33</td>
</tr>
</tbody>
</table>

external semantic knowledge by KLAPrompt approach can empower the widely-adopted pre-trained language models.

D. Ablation Study

In our proposed KLAPrompt, two components may affect the performance: Continuous Prompt and Long-answer Strategy. To explore such effects, we conduct an ablation experiment using the XNLI dataset. We first compare BERT with BERT + WSP to showcase the advantages of external semantic knowledge in WSP dataset. BERT + WSP is trained on WSP dataset with its original masked language model (MLM), and it does not use the KLAPrompt method. Experimental results demonstrate that introducing semantic information in Xinhua Dictionary can consistently improve language modeling and downstream tasks. Then we explore the effects of Continuous Prompt and Long-answer Strategy. As shown in Table III, both Continuous Prompt and Long-answer Strategy can improve performance on this Natural Language Inference dataset. In addition, the improvement brought by using Continuous Prompt or Long-answer Strategy alone is less than using the whole KLAPrompt method.

The hyper-parameter $l$ is the number of virtual tokens in the continuous prompt. To explore its impact on the performance of KLAPrompt, we test with different values of hyper-parameter $l = \{1, 2, 3, 4, 5\}$. As shown in Table IV, we can see that the performance of the model shows a trend of rising at first and then falling as $l$ increases. Especially when $l = 2$, the model has the best performance.

We also investigate the consistent improvements with different percentages of downstream training data. The experiment results in Figure 2 illustrate that the improvement is more obvious when the amount of data is smaller. In other words, KLAPrompt with semantics knowledge can benefit data-scarce downstream tasks. Because when the training data is limited, the task depends on the pre-trained language model and the additional semantics knowledge.

V. Conclusion

In this work, we propose the KLAPrompt approach to introduce semantics knowledge into pre-trained language models. What’s more, we collect a word sense prediction dataset (WSP). Extensive experiments on five Chinese NLP datasets show the effectiveness of KLAPrompt method in integrating semantic knowledge. For future work, we will infuse commonsense information, domain-specific information, and knowledge graphs into the pre-trained language models.

REFERENCES


Reinforced Multi-modal Circulant Fusion based Transformers for Rumor Detection

1st Xingang Wang
School of Computer Science and Technology
Qilu University of Technology
(Shandong Academy of Sciences)
Jinan, China
xgwang@qlu.edu.cn

2nd Xiaomin Li
School of Computer Science and Technology
Qilu University of Technology
(Shandong Academy of Sciences)
Jinan, China
xiaominli@163.com

3rd Xiaoyu Liu
School of Computer Science and Technology
Qilu University of Technology
(Shandong Academy of Sciences)
Jinan, China
XiaoyuLiuQlu@163.com

Abstract—While the popularity of social media platforms facilitates people’s communication, it also contributes to the diffusion of rumors, which may endanger social and public order. Multi-modal rumors in the form of text and images are more likely to gain public trust than traditional text-only content. As a result, the rapid identification of multi-modal rumor information in social media has emerged as a significant research focus today.

Although multi-modal rumor detection has been widely researched, we still face a challenge. We found that the current multi-modal fusion methods for this task mainly include element-wise product, element-wise sum, or even direct splicing, which may not better exploit the advantages of the complementary nature of multi-modal data. Moreover, previous studies used the fused features immediately for downstream tasks, which lacks further exploration of fused features.

To solve the above issues, we propose Reinforced Multi-modal Circulant Fusion (MCF) based Transformers for Rumor Detection (MCFbT), which considers further optimization of the fused multi-modal features. Firstly, we use two pre-trained models, ALBERT and VGG-19, to extract the textual and visual features of the posts. Secondly, we fuse the two features using MCF to obtain the initial multi-modal features. Thirdly, we enhance the multi-modal features via the cross-modal transformers. Finally, the optimized multi-modal features are reinforced using cross-modal transformers.

To solve the above issues, we propose Reinforced Multi-modal Circulant Fusion based Rumor Detection Transformers (MCFbT), which considers further optimization of the fused multi-modal features. Firstly, we use two pre-trained models, ALBERT and VGG-19, to extract the textual and visual features of the posts. Secondly, we fuse the two features using MCF to obtain the initial multi-modal features. Thirdly, we enhance the multi-modal features via the cross-modal transformers. Finally, the optimized multi-modal features are reinforced using cross-modal transformers.

Overall, in this paper, we make the following contributions:

- In the multi-modal rumor detection task, we consider for the first time allowing the individual modalities to enrich the multi-modal features by cross-modal transformers, and the optimized multi-modal features focus more on the key parts of the information.
- We evaluate MCFbT on two public real-world datasets. The results demonstrate that the model outperforms previous models.

II. RELATED WORK

In this section, previous research is briefly described in terms of both text-based rumor detection and multi-modal rumor detection.

A. Text-based Rumor Detection

Existing methods [4], [14], [15] focus mainly on extracting features from the text content of posts, which has been extensively researched in many papers on rumor detection. For
instance, Castillo et al. [4] investigate behavioral characteristics and conclude that users who have posted more messages are more likely to post truthfulness based on the posters’ previous posting history. Ma et al. [13] used TF-IDF to represent news texts and then used a recurrent neural network (RNN) to model the Weibo dataset. Chen et al. [5] proposed an RNN-based deep attention model to identify rumors by selectively learning the temporal representation of sequential posts and validated the effectiveness of RNN in the study. Potthast et al. [15] conducted several experiments using the random forest algorithm to compare the effectiveness of topic-based posts and style-based posts features for identifying rumors and found that style-based analysis only did not identify rumors. Yu et al. [26] attempted to use CNN for early rumor detection by extracting multi-scale text features through N-gram windows of varying sizes. Liao et al. [12] segmented tweets into different intervals and used a two-layer GRU network with an attention mechanism to obtain the potential features of tweets and temporal sequences.

B. Multi-modal Rumor Detection

It has been shown that images can provide effective visual information. Therefore, it is necessary to consider research based on multimodal forms in rumor detection, i.e., using text and image information to detect rumors.

In the multimodal rumor detection, Jin et al. [10] introduced neural networks to fuse different modality features of posts and attention mechanism to extract contextual relevant information for rumor detection. Wang et al. [24] combined the idea of adversarial learning to propose the EANN model, which adds the sub-task of event classification to the original classification task and induces the model to learn event-independent features to improve the generalization ability of the model. Khattar et al. [11] proposed an end-to-end multi-modal variational self-encoder model (MVAE) that learns the potential distribution of multi-modal features to explore the distribution pattern of rumor. Chen et al. [7] proposed the CAFE rumor detection model, which can adaptively aggregate unimodal and cross-modal correlations and thus improve task accuracy. Qi et al. [16] innovatively extract visual entities (e.g., celebrities and landmarks) to understand the high-level semantics of post-related images and then model the inconsistency and mutual enhancement of multimodal entities with the help of visual entities. Dhawan et al. [9] proposed GAME-ON, an end-to-end trainable framework based on a graph neural network that allows granularity interactions within and across different modalities to learn more robust data.

Unlike all the above work, we consider feature enhancement after the fusion of images and text when identifying rumors on social media.

III. METHODOLOGY

In this section, we mainly introduce the proposed model MCFbT in detail. As shown in Fig. 1, MCFbT has four parts:

- **Multi-modal Feature Extractor.** A pre-trained model ALBERT, is used to model the text and obtain the text features, while the VGG-19 network is used to extract the visual feature of images.

- **Multi-modal Feature Fusion.** The MCF method is used to fuse visual and textual features to obtain the initial multi-modal features.

- **Multi-modal Feature Enhancement.** To obtain a more expressive feature representation, we use an architecture consisting mainly of cross-modal converters to optimize the initialized multimodal features.

![Fig. 1. The overall framework of MCFbT.](image)

A. Multi-Modal Feature Extractor

To accurately model the semantic and contextual meaning of words, the paper uses the ALBERT [8] model with 12 encoder layers for text feature extraction. ALBERT ensures the quality of feature extraction while reducing parameters and improving training speed. The model is used to obtain features of the complete text, $S_t \in \mathbb{R}^{1 \times d_t}$, where $d_t$ denotes the dimensionality of the text features obtained from the ALBERT, we do not fine-tune the ALBERT weights, so the ALBERT network’s weights are frozen.

We refer to previous research [17], [22], [27], and choose to use the VGG-19 [19] network to extract visual features from images. This is a 19-layer VGGNet pre-trained on the ImageNet dataset [19] to obtain visual feature vectors, $S_v \in \mathbb{R}^{1 \times d_v}$ where $d_v$ is the visual feature dimension output by VGG-19. The pre-trained model is fixed during training, and
the weights of VGG-19 are not fine-tuned, and the weights of the VGG network are frozen.

Since it is possible that \( d_t \neq d_v \), we design that a fully connected layer is added to ensure the same dimensionality \( d \) as the two features, as follows:

\[
\begin{align*}
\hat{S}_t &= \text{ReLU}(S_t \times W_t + b_t) \\
\hat{S}_v &= \text{ReLU}(S_v \times W_v + b_v)
\end{align*}
\]

(1) \( (2) \)

where \( W_t \in \mathbb{R}^{d_t \times d} \) and \( W_v \in \mathbb{R}^{d_v \times d} \) are the weight matrices, and \( b_t \) and \( b_v \) are the bias terms.

### B. Multi-modal Feature Fusion

The paper adopts the MCF method to achieve feature fusion. The method uses the newly defined interaction operation to complete the fusion after transforming the feature vector into circulant matrices [25].

We use the projection vector \( X \in \mathbb{R}^{d_t} \), and \( Y \in \mathbb{R}^{d_v} \) to construct circulant matrix \( A \in \mathbb{R}^{d_t \times d_t} \) and \( B \in \mathbb{R}^{d_v \times d_v} \).

\[
A = \text{circ}(\hat{S}_t)
\]

\[
B = \text{circ}(\hat{S}_v)
\]

where \( \text{circ}(d) \) denotes converting \( d \) to a circulant matrix.

Then the projection vector and each row vector of the circulant matrix do an element-wise product. The algorithm is as follows:

\[
M = \frac{1}{d} \sum_{i=1}^{d} a_i \odot \hat{S}_t
\]

\[
N = \frac{1}{d} \sum_{i=1}^{d} b_i \odot \hat{S}_v
\]

(5) \( (6) \)

where \( a_i \in \mathbb{R}^{d_t} \) and \( b_i \in \mathbb{R}^{d_v} \) are the row vectors of the circulant matrix \( A \) and circulant matrix \( B \) respectively, \( \odot \) denotes the operation of the element-level product. Crucially, this process does not introduce new parameters.

Finally, through the projection matrix \( W_t \in \mathbb{R}^{d_t \times d_v} \), we convert the element-level sum vectors of \( M \in \mathbb{R}^{d_t} \) and \( N \in \mathbb{R}^{d_v} \) into the target vectors \( \hat{R}_f \in \mathbb{R}^{d_v} \), the multi-modal features.

### C. Multi-modal Feature Enhancement

We obtain textual features \( \hat{S}_t \), visual features \( \hat{S}_v \), and multi-modal features \( \hat{S}_f \) through the above two subsections, all of which have dimensions of \( d \). Below we will introduce the multi-modal features optimization architecture based on transformers [18].

#### Temporal Convolutions

To ensure that each element in the input sequence has a sufficient understanding of its neighbors, we pass the input sequence through a 1D temporal convolutional layer:

\[
R_{t,v,f} = \text{Conv1D} \left( \hat{S}_{t,v,f}, k_{t,v,f} \right) \in \mathbb{R}^{T_{t,v,f} \times d}
\]

(7)

where \( k_{t,v,f} \) are the sizes of the convolutional kernels for modalities \((t, v, f)\), and \( d \) is a common dimension.

#### Positional Embedding

In order to enable multi-modal feature sequences to carry temporal information, we add positional embedding \((PE)\) to \( R_f \):

\[
\hat{R}_{t,v,f} = R_{t,v,f} + PE(T_{t,v,f}, d)
\]

(8)

where \( PE(T_{t,v,f}, d) \in \mathbb{R}^{T_{t,v,f} \times d} \) calculate the (fixed) embedding vectors for each position index, and \( R_{t,v,f} \) is a low-level location-aware feature generated for multi-modal features.

#### Cross-modal Transformers

We refer to the cross-modal transformer designed in previous studies [21], which enables one modality to receive information from another modality. We assume that the textual information \((t)\) is passed to the multi-modal information \((f)\), denoted by \( t \rightarrow f \). The complete calculation process is as follows:

\[
\hat{R}^{[0]}_{t \rightarrow f} = Z^{[0]}_f
\]

(9)

\[
\hat{R}^{[i]}_{t \rightarrow f} = \text{CM}^{[i],\text{mul}} \left( \text{LN} \left( \hat{R}^{[i-1]}_{t \rightarrow f} \right), \text{LN} \left( \hat{R}^{[0]}_{t \rightarrow f} \right) \right) + \text{LN} \left( \hat{R}^{[i-1]}_{t \rightarrow f} \right)
\]

(10)

where \( \theta \) is a positionwise feed-forward sublayer parametrized by \( \theta \), and \( \text{CM}^{[i],\text{mul}} \) means a multi-head version of \( \text{CM} \rightarrow f \) [21] at layer \( i \). \( LN \) means layer normalization [2].

\( \hat{R}^{[i]}_{t \rightarrow f} \) follows the same steps as described above. Then, we concatenate the outputs from the cross-modal transformers to obtain the final multi-modal features \( \hat{R}_f = \left[ \hat{R}^{[0]}_{t \rightarrow f}; \hat{R}^{[D]}_{v \rightarrow f} \right] \).

\( \hat{R}_f \in \mathbb{R}^{T_f \times 2d} \).

#### D. Rumor Detection

We project the multi-modal feature vector \( \hat{R}_f \) into the target space of whether it is a rumor or not using a fully connected layer with softmax activation and obtain the probability distribution \( p \):

\[
p = \text{softmax} \left( W \hat{R}_f + b \right)
\]

(12)

where \( W \) is the weight matrix and \( b \) is the bias term.

To calculate the classification loss, we use the cross-entropy loss as follows:

\[
L_p = - [y \log p_0 + (1 - y) \log p_1]
\]

(13)

where \( y \in \{0, 1\} \) denotes the ground-truth label.

### IV. Experiments

#### A. Datasets

To assess the effectiveness of the MCFbT, we conducted experiments on two real-world datasets, which are collected from Twitter and Weibo, respectively. The Twitter dataset was released for Verifying Multimedia Use task at MediaEval [3]. It is divided into two parts, the development set and the test set, with a ratio of 7:3, and there is no overlap between them. We use the development set for training and the test set for evaluation to compare baselines fairly. The Weibo dataset is collected by Jin et al. [10], basic situation of which is similar to the Twitter. The two datasets use images and text, so we remove posts without any text or images. Table II shows the statistics of the two datasets.
TABLE I
THE RESULTS OF DIFFERENT METHODS ON TWO DATASETS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Accuracy</th>
<th>Rumour</th>
<th>Non-rumour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Twitter</td>
<td>Text-Only</td>
<td>0.706</td>
<td>0.648</td>
<td>0.540</td>
</tr>
<tr>
<td></td>
<td>Image-only</td>
<td>0.596</td>
<td>0.695</td>
<td>0.518</td>
</tr>
<tr>
<td></td>
<td>VQA</td>
<td>0.631</td>
<td>0.765</td>
<td>0.509</td>
</tr>
<tr>
<td></td>
<td>NeuralTalk</td>
<td>0.610</td>
<td>0.728</td>
<td>0.504</td>
</tr>
<tr>
<td></td>
<td>att-RNN</td>
<td>0.682</td>
<td>0.780</td>
<td>0.615</td>
</tr>
<tr>
<td></td>
<td>EANN</td>
<td>0.719</td>
<td>0.642</td>
<td>0.474</td>
</tr>
<tr>
<td></td>
<td>MVAE</td>
<td>0.745</td>
<td>0.801</td>
<td>0.719</td>
</tr>
<tr>
<td></td>
<td>CAFE</td>
<td>0.806</td>
<td>0.807</td>
<td>0.799</td>
</tr>
<tr>
<td></td>
<td>MCFbT</td>
<td>0.842</td>
<td>0.830</td>
<td>0.863</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Accuracy</th>
<th>Rumour</th>
<th>Non-rumour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weibo</td>
<td>Text-Only</td>
<td>0.804</td>
<td>0.800</td>
<td>0.860</td>
</tr>
<tr>
<td></td>
<td>Image-only</td>
<td>0.633</td>
<td>0.630</td>
<td>0.500</td>
</tr>
<tr>
<td></td>
<td>VQA</td>
<td>0.736</td>
<td>0.797</td>
<td>0.634</td>
</tr>
<tr>
<td></td>
<td>NeuralTalk</td>
<td>0.726</td>
<td>0.794</td>
<td>0.613</td>
</tr>
<tr>
<td></td>
<td>att-RNN</td>
<td>0.788</td>
<td>0.862</td>
<td>0.686</td>
</tr>
<tr>
<td></td>
<td>EANN</td>
<td>0.816</td>
<td>0.820</td>
<td>0.820</td>
</tr>
<tr>
<td></td>
<td>MVAE</td>
<td>0.824</td>
<td>0.854</td>
<td>0.769</td>
</tr>
<tr>
<td></td>
<td>CAFE</td>
<td>0.840</td>
<td>0.855</td>
<td>0.830</td>
</tr>
<tr>
<td></td>
<td>MCFbT</td>
<td>0.870</td>
<td>0.856</td>
<td>0.875</td>
</tr>
</tbody>
</table>

TABLE II
THE STATISTICS OF THE REAL-WORLD DATASETS.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Label</th>
<th>Number</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter</td>
<td>fake</td>
<td>7021</td>
<td>12995</td>
</tr>
<tr>
<td></td>
<td>real</td>
<td>5974</td>
<td></td>
</tr>
<tr>
<td>Weibo</td>
<td>fake</td>
<td>4749</td>
<td>9528</td>
</tr>
<tr>
<td></td>
<td>real</td>
<td>4779</td>
<td></td>
</tr>
</tbody>
</table>

B. Baseline Model
To verify the validity of this model, we selected the baseline model from the following two models: uni-modal models, multi-modal models.

(i) Uni-modal models
- Text-only. A model that uses only text information and extracts textual features by ALBERT.
- Image-Only. A model that uses only image information and extracts visual features by VGG-19.

(ii) Multi-modal models
- VQA. VQA [1] is a model for answering questions about a given image. For comparison with MCFbT, the VQA is designed for a binary classification task.
- NeuralTalk. NeuralTalk [23] is a deep recurrent framework for image caption. To adapt rumor detection task, the model feature is averaged over the RNN output at each time step.
- att-RNN. The att-RNN [10] uses an attention mechanism to combine textual, visual and social context features. In the paper, the part dealing with social environment information is removed from the experiments.
- EANN. EANN is the Event Adversarial Neural Networks proposed in [24]. The framework is to guide the model to learn event-independent multi-modal features by introducing an event classifier as a secondary task.
- MVAE. MVAE [11] is a multi-modal model based on Variational Autoencoder. The spliced features are encoded as an intermediate expression for reconstructing input features and rumor classification.
- CAFE. CAFE [6] is a multi-modal rumor detection method with fuzzy perception. The model can wisely and adaptively aggregate uni-modal features and cross-modal correlations.

C. Results and Analysis
Table I shows the results of baselines and our proposed model on two datasets. The experimental results show that our proposed method outperforms the baseline. The following are some specific observations:

- Text-based method outperforms images-based approach, proving that text contains more information than images.
- The multi-modal models outperform the uni-modal model, showing that multi-modal methods generally outperform uni-modal based methods, demonstrating the superiority of multi-modal Features.
- Among the multi-modal models tested in the baseline, MCFbT performed the best, indicating that the model can accurately capture more effective multi-modal features to detect rumors.

To demonstrate the importance of each component, we perform ablation analysis on the MCFbT model and experiment on Twitter and Weibo datasets, the following variants of our model are designed for comparison:

- Base: the most basic model composed of the ALBERT, the VGG-19 and the MCF.
- Base-tf: models augmented with multi-modal features rely only on textual features.
Base-vf: models augmented with multi-modal features rely only on visual features.

MCFbT: The whole model MCFbT with all components.

As shown in Fig. 2, both Base-tf and Base-vf are more effective than the Base, and the overall model MCFbT results are higher than the Base-tf and the Base-vf, indicating that each component is necessary and effective. In particular, we find that Base-tf is better than Base-vf on both datasets, suggesting that textual features cover richer semantic information to help us detect rumors.

V. CONCLUSION

In this paper, we propose MCFbT, Reinforced Multi-modal Circulant Fusion based Transformers for Rumor Detection. In this task, we consider using textual and visual features to optimize multi-modal features after fusion via cross-modal Transformers. Compared with the traditional method, the model makes it pays more attention to vital information and relatively reduces the influence of noise. We conduct extensive experiments on two real-world datasets, the results demonstrate the effectiveness of our proposed method. In the future, we plan to conduct research from a visual information perspective to contribute to rumor detection with potentially better results.

REFERENCES

A Medical Question Classification Approach Based on Prompt Tuning and Contrastive Learning

Qian Wang  
School of Computer Science and Information Engineering  
School of Artificial Intelligence  
Hubei University  
Wuhan, China  
wqmaster@stu.hubu.edu.cn

Cheng Zeng*  
School of Computer Science and Information Engineering  
School of Artificial Intelligence  
Hubei University  
Wuhan, China  
zc@hubu.edu.cn

Yuixin Liu  
School of Computer Science and Information Engineering  
School of Artificial Intelligence  
Hubei University  
Wuhan, China  
1298740592@qq.com

Peng He  
School of Computer Science and Information Engineering  
School of Cyber Science and Technology  
Hubei University  
Wuhan, China  
penghe@hubu.edu.cn

Abstract—COVID-19 has profoundly impacted people’s lives, and people are more concerned about medical and health issues, so it is essential to design an efficient method for classifying medical questions. Fine-tuning paradigms based on pre-trained language models have proven effective in recent years. However, PLMs based on fine-tuning paradigms are poorly robust, and there is a gap between the pre-training phase and the downstream task form, resulting in PLMs that cannot use the rich latent knowledge in downstream tasks. We propose a medical question classification method that combines prompt fine-tuning and contrastive learning and uses the large-scale knowledge graph enhancement model ERNIE 3.0 as a feature extractor to address both problems. Our approach utilizes an additional prompt template to enable PLM to unleash the potential in specific tasks and uses a contrast sample strategy to alleviate the problem of confusable samples that are difficult to distinguish. Experiments on a medical question classification dataset show that the method achieves an accuracy of 93.65 percent, with better metrics than recent work.

Keywords—medical question classification ; contrastive learning; prompt tuning; ERNIE 3.0

I. INTRODUCTION

The COVID-19 pandemic has inflicted immense damage worldwide, revealing the fragility of global healthcare delivery systems. The internet is inundated daily with concerns about the health consequences of COVID-19 sequelae[1]. Given this context, accurate medical question classification is of paramount interest. However, due to the scarcity of professional medical guides and ambiguous phrasing of medical inquiries, efficiently handling user-raised medical questions is challenging. Precise categorization of medical health questions is thus crucial.

In this paper, we ground the Medical Question Classification Task (MQC), where the goal is to accurately classify medical questions. Currently, MQC is defined as a single-label multi-class prediction problem[2], and the implementation methods are mainly classified into statistical-based machine learning methods and neural network-based deep learning methods. In recent years, with the arrival of the artificial intelligence wave, deep learning has been the most widely used method in MQC. However, there are drawbacks of insufficient extraction of problematic features and inability to utilize the corpus. Pre-trained language models solve this problem. Rasmy et al. [1] developed MedBERT, a medical knowledge-aware BERT model based on medical knowledge, which can give larger weights to medical concept words in the problem text and achieve good classification under fine-tuning. He et al. [4] proposed a BERT model to extend long and multi-type texts by integrating medical knowledge graphs, which also achieved better classification results under fine-tuning. But an important issue is overlooked: the target task for the initial training of PLMs does not match the downstream task, resulting in the inability to unleash the potential of PLMs in a specific task.

To address the problems of the medical question classification task, we propose a medical question classification method based on prompt tuning and contrastive learning. We adopt ERNIE 3.0[5] as a feature extractor to support the construction of prompt learning. ERNIE 3.0, a PLM with a large-scale knowledge base, performs surprisingly well in natural language processing tasks. Moreover, we introduce the contrastive learning approach to the current prompt-based fine-tuning task. Our contributions can be summarized as follows:

- For the medical problem classification task, we propose a model that combines prompt learning and supervised contrastive sample learning. We construct a prompt template as part of the original input to assist ERNIE 3.0 in achieving excellent classification results in our experiments.
- In the training process, we adopt the comparative sample learning strategy to alleviate the problem of insufficient medical samples.

II. RELATED WORK

A. Knowledge Enhanced Pre-Training model ERNIE 3.0

PLMs such as BERT[6], RoBERTa[7], and GPT[8] perform impressively on various natural language understanding and generation tasks. However, most models learn only on corpora and lack experience in aspect-based tasks. Therefore, this approach performs poorly in solving downstream language understanding tasks. In this paper, we use a pre-trained language model containing billions of parameters for Chinese knowledge enhancement trained by PaddlePaddle[9]. It fuses autoregressive and self-coding networks and was trained on a 4TB Chinese corpus and a large-scale knowledge graph corpus. As shown in Figure 1, ERNIE 3.0 uses multilayer Transformer-XL[10] as the basic structure. In ERNIE 3.0 architecture, the universal representation module is used to extract initial semantic features, and these parameters are shared in the understanding and generation tasks. The task-specific representation plays the role of task-specific semantic feature extraction, and the task-specific target learns the parameters.
B. Prompt-based Tuning

Pre-trained language models based on prompt turning have been shown to be effective for different NLP tasks [11]. Prompt universal process, as shown in Figure 2: identify PLM tasks, design prompt engineering, and answer engineering. PET [12] is a traditional prompt learning method requiring less annotated data than fine-tuning. Gao et al.[13] chooses to optimize prompt token embedding based on prompt tuning and proposes a small-sample fine-tuning method based on PLM, which also performs well in scenarios with few resources. Liu et al.[14] proposes P-tuning to insert optimized pseudo prompt tokens at the input side to find knowledge templates in continuous space automatically.

C. Contrastive Learning

In many practical applications, researchers face challenges in collecting large-scale data and achieving precise labeling. To overcome these issues, researchers have proposed several methods for deep feature extraction from unlabeled or weakly labeled samples. The fundamental concept underlying contrastive learning is to measure the similarity of sample pairs in the feature space. By leveraging data sample labeling information, this technique pulls similar samples closer together in the feature space during training, while increasing the distance between dissimilar samples. SimCSE [15] implements textual sentence-level semantic representation based on contrastive learning and dropout, unsupervised SimCSE uses dropout to construct positive and negative samples and supervised SimCSE constructs positive and negative samples with the help of contradictory labels contained in the NLI dataset. However, in supervised situations, the scarcity of NLI data is not conducive to enhancing the semantic representation of sentences. Khosla et al.[16] introduced contrastive learning to the supervised scenario and achieved significant results in text classification task.

III. APPROACH

In this section, we discuss our proposed ERNIE 3.0 medical question classification method based on prompt-turning and contrastive learning. The model's primary steps are prompt template construction and contrastive sample learning, as shown in Figure 3. The steps are detailed as follows:

A. Prompt Engineering

Prompt template construction: In this paper, the prompted approach treats the classification task as a masked language modeling (MLM) problem. The training data for our classification task is composed of \( D = \{ x, y \} \), where \( x \) is the set of input texts and \( y \) is the set of labels corresponding to the texts, with each text corresponding to a label. We use specific [Mask] token and template \( T \) to construct the input instance \( x_p \) and set the template \( T \) as "This is a [Mask][Mask] intention" for this task, while we define a set of characters \( P, p \in P \) to be used for filling the prompt template.

ERNiE3.0

Constrastive Simple Learning Module

\[ \begin{align*}
L_{CL} & \quad \text{CL loss} \\
L_{CE} & \quad \text{CE loss} \\
L_{KL} & \quad \text{KL divergence} \\
E_{cl} & \quad \text{classification loss} \\
E_{cb} & \quad \text{classification boosting loss} \\
E_{ce} & \quad \text{contrastive loss} \\
L_{CLS} & \quad \text{CL loss} \\
L_{CE} & \quad \text{CE loss} \\
L_{KL} & \quad \text{KL divergence} \\
E_{cl} & \quad \text{classification loss} \\
E_{cb} & \quad \text{classification boosting loss} \\
E_{ce} & \quad \text{contrastive loss} \\
\end{align*} \]

Figure 3. The framework of ERNiE 3.0-CL on Prompt turning.
Combination of input and template: Before prompt-based contrastive learning, we need to convert $x$ to $x_{\text{prompt}}$, as shown in Figure 3, where we place the template in front of the sentence, and the final input to the model is:

$$x_{\text{prompt}} = \{[CLS], T, [SEP], x\}$$

Answer Mapping: We use $h \in \mathbb{R}^{L \times d}$ as the [Mask] hidden layer state of the PLM, where $L$ represents the length of $x_{\text{prompt}}$, $d$ represents the hidden layer size. To map hidden vectors to relational labels, we define a verbalizer $V_o$ as a mapping method. The probability score of category $i$ is:

$$p(y_i|x) = p(V_o|x_{\text{prompt}}) = \frac{\exp(W_i \cdot h)}{\sum_{j} \exp(W_j \cdot h)}$$

where $W$ is the static embeddings of the label prompt token, with size $l \times d$.

We build prompt input sentences and make the model predict the label with the highest correlation to [mask]. Consequently, our classification task is transformed into cloze-style task.

B. Contrastive Simple Learning Module

Large-scale labeled data collection is complex and expensive in many application situations. To address this problem, scholars have proposed various methods for engineering features on unlabeled data. Many scholars have proved self-supervised and supervised contrastive learning methods to improve the effectiveness of sample features and the robustness of models. However, self-supervised contrastive learning does not make effective use of supervised information, and it is not applicable to the question classification task. In this paper, we use the supervised contrastive learning strategy to expand the features; referring to the previous study [16], we define the $i$-th sample as the target sample, the $j$-th sample as the positive sample, and the remaining N-2 samples as the negative samples. The following contrastive loss is defined for supervised tasks:

$$L_{CL} = - \frac{1}{N} \sum_{i \in \mathcal{P}(i)} \frac{1}{|\mathcal{P}(i)|} \sum_{j \in \mathcal{P}(i)} \log \frac{\exp(z_i \cdot z_j / \gamma)}{\sum_{k \in \mathcal{L}_h} \exp(z_i \cdot z_k / \gamma)}$$

where $\mathcal{P}(i)$ is the set of indices of all positive simples in the mini-batch, $\gamma$ is a temperature coefficient, $N$ is the number of batch size, $z$ is the normalized representation of the ERNIE 3.0 feature vectors $h$ based on the prompt fine-tuning method. Although this approach effectively compares positive and negative samples, we still need a cross-entropy loss to fine-tune the classifier:

$$L_{CE} = - \frac{1}{N} \sum_{i \in \mathcal{P}(i)} \log \frac{\exp(z_i \cdot z_j)}{\sum_{a \in \mathcal{A}} \exp(z_i \cdot z_a)}$$

Finally, the total model loss is:

$$L = L_{CE} + \alpha \cdot L_{CL}$$

where $\alpha$ is the hyperparameter used to control the contrastive learning and cross-entropy loss.

IV. EXPERIMENTS

A. Experimental dataset and settings

In this paper, we use a crawler tool to crawl texts from mainstream medical websites to construct a medical question dataset for experiments, using manual annotation. Based on the crawled data, we integrated and labeled 5 categories: disease diet, seasonal diet, sports and fitness, weight loss and beauty, and dietary contraindications.

<table>
<thead>
<tr>
<th>Label</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>seasonal diet</td>
<td>The principles of winter health care.</td>
</tr>
<tr>
<td>disease diet</td>
<td>What to eat after injury fracture.</td>
</tr>
<tr>
<td>dietary contraindications</td>
<td>People who should not eat watermelon.</td>
</tr>
</tbody>
</table>

The experiment in this paper is a multiclass classification task, and we use the accuracy rate and F1 value as the main evaluation metric.

B. Implementation Details

The model is trained using AdamW[17] gradient descent algorithm, with batch size of 64, maximum rounds of 50, and initial learning rate of 2E-5. We set dropout to 0.1 in all layers of the model. The hyperparameters in the contrastive learning module are the contrastive learning loss fusion factor $\alpha$, the temperature coefficient $\gamma$. After several experimental comparisons, we obtained the optimal model parameters with partial hyperparameters of $\alpha$ set to 0.01 and $\gamma$ set to 0.1.

C. Baseline models and approaches

We compare with several baselines in related work, using BERT and RoBERTa as pre-trained language models. We also compare the fine-tuned based training method of PLMs with the integration of other advanced methods SimCSE, R-drop.

V. ANALYSIS OF EXPERIMENTAL RESULTS

To evaluate the effectiveness of the proposed method, we used the proposed method to compare each baseline and conducted experiments on the medical question dataset, and the results are shown in TABLE II., from which we can obtain the following conclusions.

- BERT and RoBERTa have slight difference in experimental effect, ERNIE 3.0 introduces large-scale knowledge graph in pre-training model to get more text features and better classification effect.
- TABLE II. shows that our method works well for prompt-based methods. The Prompt-Tuning-based ERNIE 3.0-CL method proposed in this paper improves the accuracy by 0.8 percentage points and the F1 value by 0.6 percentage points. Its performance is also better than PLMs using the fine-tuning paradigm.
- All ERNIE 3.0 fusion models are more effective than the baseline ERNIE 3.0. Our proposed ERNIE 3.0-CL compares better with other methods of ERNIE 3.0 fusion with contrastive learning, ERNIE 3.0-Rdrop, and ERNIE 3.0-SimCSE. We believe this is because contrastive learning learns a more generalized representation from positive and negative case samples; contrastive learning module does improve the model's ability to discriminate between negative samples.

634
TABLE II. THE COMPARATIVE RESULTS OF THE EXPERIMENTS

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>0.9213</td>
<td>0.924</td>
</tr>
<tr>
<td>BERT-Rdrop</td>
<td>0.9228</td>
<td>0.9284</td>
</tr>
<tr>
<td>BERT-SimCSE</td>
<td>0.9259</td>
<td>0.9306</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>0.9265</td>
<td>0.9303</td>
</tr>
<tr>
<td>RoBERTa-Rdrop</td>
<td>0.9276</td>
<td>0.9306</td>
</tr>
<tr>
<td>RoBERTa-SimCSE</td>
<td>0.927</td>
<td>0.9318</td>
</tr>
<tr>
<td>RoBERTa-CL</td>
<td>0.9291</td>
<td>0.9323</td>
</tr>
<tr>
<td>ERNIE3.0</td>
<td>0.9281</td>
<td>0.9334</td>
</tr>
<tr>
<td>ERNIE3.0-Rdrop</td>
<td>0.9286</td>
<td>0.9314</td>
</tr>
<tr>
<td>ERNIE3.0-SimCSE</td>
<td>0.9307</td>
<td>0.9327</td>
</tr>
<tr>
<td>Prompt-tuning ERNIE3.0-CL</td>
<td>0.9365</td>
<td>0.9391</td>
</tr>
</tbody>
</table>

VI. HYPERPARAMETER INFLUENCE

In supervised contrastive learning, there are some hyperparameters that can affect the model's performance and training process. In order to explore the effectiveness of contrastive learning, we set different fusion ratio $\alpha$ and temperature coefficient $\gamma$ to control contrastive learning loss. The correct setting of temperature parameter can make the model learn hard negatives better. As shown in Figure 4, after experimental comparison of the two hyperparameters we set, when the temperature coefficient is set to 0.1, and the fusion ratio is set to 0.01, our model gets the best result.

In this paper, we design a novel supervised contrastive learning method for medical question classification using ERNIE 3.0 pre-trained language model and prompt fine-tuning. The classification problem is transformed into a completion fill-in-the-blank problem through a manual prompt template to learn words mapped to labels and using a contrastive sampling module to pull in similar samples and push negative samples from different classes. We explore the efficiency of our model on the medical question classification task and outperform recent work in experimental tasks. However, our proposed model has limitations: The prompt template uses manual templates, which have more parameters to optimize and are prone to overfitting. In our subsequent work, we will take the automatic generation of prompt templates and the optimization of sampling strategies as our direction and try experimenting with different tasks.

VII. CONCLUSION

In this paper, we design a novel supervised contrastive learning method for medical question classification using ERNIE 3.0 pre-trained language model and prompt fine-tuning. The classification problem is transformed into a completion fill-in-the-blank problem through a manual prompt template to learn words mapped to labels and using a contrastive sampling module to pull in similar samples and push negative samples from different classes. We explore the efficiency of our model on the medical question classification task and outperform recent work in experimental tasks. However, our proposed model has limitations: The prompt template uses manual templates, which have more parameters to optimize and are prone to overfitting. In our subsequent work, we will take the automatic generation of prompt templates and the optimization of sampling strategies as our direction and try experimenting with different tasks.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No. 62102136), the Key R & D projects in Hubei Province (No.2021BA1A188, 2021BA1A184, 2022BA044), the Science and Technology Innovation Program of Hubei Province (No.2020AEA008).

REFERENCES


(a) Loss function fusion ratio

(b) Temperature coefficient

Figure 4. Hyperparameter comparison.
An intelligent question answering system based on RoBERTa-WWM under home appliance knowledge graph

Xiaoyan Gao, Ji Chen, Xiangqian Ding*, Shuo Shi, Junhua Jia, Xiaowei Xin
Faculty of Information Science and Engineering, Ocean University of China, Qingdao, China
Email: gxy9924@stu.ouc.edu.cn, 1553316917@qq.com, Dingxq1995@vip.sina.com, ss@ouc.edu.cn, jiajunhua@stu.ouc.edu.cn, xinxiaowei91@163.com

Abstract—Recently, the intelligent question answering (QA) system for home appliances has attracted wide attention in China because it can provide users with reasonable suggestions in time. However, the fuzzy boundaries of the Chinese "word groups" and the absence of a unified standard for Chinese question classification result in low efficiency of QA, which has hindered the development of the system for a long time. In this work, we developed a QA model based on RoBERTa-WWM to answer questions about home appliances in specific fields. More importantly, the RoBERTa-WWM-BiLSTM-CRF named entity recognition model and RoBERTa-WWM-TextCNN question semantic classification model were constructed to parse the correct semantics of questions from users, which provided algorithm support for the home appliance QA system. The results showed that the method outperformed baselines, and the QA model showed high interpretability and good performance.

Keywords—knowledge graph question answering; home appliance; named entity recognition; semantic classification

I. INTRODUCTION

The rapid development of e-commerce in China has recently led to a sharp increase in people shopping online. Since 2020, online sales of home appliances have accounted for over 50% and are expected to continue to increase [1]. Surging consumption, product diversification, and rapid update result in a severe imbalance between the supply and demand of consumer services. The intelligent question answering (QA) system based on the home appliance knowledge graph (KG) can provide timely and uninterrupted service, freeing customer service from answering many repetitive mechanical questions. Therefore, it can lower the service cost and improve the shopping experience. Generally, all service platforms can use the QA system to optimize service [2], indicating its broad prospects in the application.

The current QA system in the field of e-commerce is absent of flexibility because it is mainly based on rule matching or template matching and requires manual template configuration. For example, a customer service robot system developed by Li et al. [3] obtained the vector representation of the question and the answer by CNN training, followed by returning the corresponding answer by similarity matching. A cosmetic QA system based on KG built by Xue et al. [4] used a BERT-CRF model to identify the relevant entities of the question and then offered an answer according to a preset question template. In the field of home appliances, the low efficiency of template updating by hand can hardly keep up with the rapid update of products. The QA system based on semantic parsing is the solution to the abovementioned problem. The entity and intention of the question are determined by semantic parsing, followed by the automatic construction of the query statement. Then, the answer is found in the knowledge base, indicating that automated QA is realized.

However, some problems still need to be solved in the semantic parsing of home appliance questions. Firstly, the data on home appliances contain a lot of heterogeneous information combining numbers and letters, which makes it challenging to extract uniformly. Secondly, the question corpus of home appliances is relatively limited. Finally, Chinese word groups have fuzzy boundaries, unlike English ones with delimiters to mark the boundary [5]. The fundamental element of semantic expression in Chinese is a word group rather than a single word.

Herein, a RoBERTa-WWM Chinese pre-training language model was introduced to solve the abovementioned problems. The model was trained on a large-scale Chinese dataset and combined with the whole word masking (WWM) technology. It could effectively learn the semantic information of Chinese words and word groups, which was suitable for Chinese NLP tasks. In addition, a QA method based on the RoBERTa-WWM was proposed for the QA task of home appliances. RoBERTa-WWM-BiLSTM-CRF named entity recognition (NER) model and RoBERTa-WWM-TextCNN question semantic classification model were constructed in the method. They could analyze the correct semantics of questions to provide algorithm support for the home appliance QA system.

II. METHODOLOGY

A. Design of Automated QA Process

The automated QA mainly comprises question semantic parsing, retrieval of the Neo4j graph database [6], and answer generation (Fig. 1). The former contains NER and question semantic classification, which are used for entity extraction and classification of questions from users to understand query intentions. Based on the results obtained, the middle constructs Cypher query statements and then queries the corresponding results in the graph database. The latter will sort the results

This work was financially supported by the National Key R&D Program of China (2022YFB3305804).

DOI reference number: 10.18293/SEKE23-195
*Corresponding Author
queried by Cypher, match the question, and return the answer with the highest matching degree. For an empty query result, the corresponding prompt information will be returned.

![Diagram](image)

Fig. 1 Design of automated QA process

**B. Question Corpus Generation and Data Preprocessing**

Question corpus was automatically generated to construct the corpus dataset based on crawling the data of e-commerce websites because rich question corpus data can improve the accuracy of the training model. Then, the BIO markup tool based on a dictionary was used for data preprocessing in the NER task. Finally, the question classification method was constructed for the semantic classification task.

1) **Question corpus generation based on the home appliance KG**: A method of question corpus generation based on template rules was designed according to the features of the home appliance questions, as shown in Fig. 2.

![Diagram](image)

Fig. 2 A method of question corpus generation

a) The information on the entity (home appliance model, company name, and brand name), property (home appliance energy efficiency level, size, weight, and power), and relation (company and brand; home appliance and brand) obtained from the home appliance KG was used as base data to generate the question corpus.

b) After analyzing the question structure form and designing the question template rule, entity data of different types in the KG were generated as questions according to the rules. Extending the question into a natural language question can make every entity label have a relevant relation or property label. Meanwhile, there is also no semantic ambiguity in the extended question. Subject replacement and synonym conversion were used to generalize and generate more training corpus to ensure the corpus data's diversity and accuracy because questions may cover many question-setting scenarios. Meanwhile, some mood particles or transitional words can be added to the question, which makes the corpus richer.

c) Based on the question template, the entity, relation, and property in the KG are populated by the rules to achieve the automatic generation of the question corpus.

2) **BIO sequence labeling of question corpus**: NER usually contains entity boundary recognition and entity type determination. The NER task of Chinese, a pictograph, is more challenging than that of some pinyin scripts, such as English. More precisely, "word group" is a vague concept in Chinese, and identifiers for marking word group boundaries, such as spaces in English, are also absent [5]. In addition, Chinese does not have morphological indications such as letter case in English. Therefore, the task of Chinese NER is usually regarded as a sequence labeling question.

Sequence labeling can be divided into raw labeling and joint segmentation and labeling depending on the difference in tag granularity. The former labels each word, which can be regarded as its direct classification, while the latter indicates the same label for consecutive words. Joint segmentation and labeling were used in this work because the entity consists of several words. Compared with raw labeling, there may be dependencies between adjacent word labels in the joint segmentation and labeling. Therefore, BIO labeling is generally used to convert joint segmentation and labeling into raw labeling. BIO labeling can convert each label with a cross-word in the joint segmentation and labeling into two new labels. For example, the "品牌 (brand)" entity label "BRAND" was converted into "B-BRAND" and "I-BRAND". "B" and "I" refer to the initial word and the following word of the entity, respectively, while "O" represents a word that does not belong to any of the predefined word fragment types. Some examples of labeling are shown in Table I.

<table>
<thead>
<tr>
<th>Table I. Examples of BIO labeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>美</td>
</tr>
<tr>
<td>B-BRAND</td>
</tr>
</tbody>
</table>

A corpus BIO labeling tool was prepared based on a dictionary. The dictionary of the required labeling entity could be made successfully according to the entity information from the KG. The data of each line consist of an entity name and a label, separated by spaces. After reading the question corpus, the sentence of each line was split into a single word or character to generate a labeling result set with the same length as the corpus, which was filled with an "O" label by default. Then, the entity matching the dictionary in the corpus was set to the corresponding labels after traversing the labeled entities in the dictionary. The corpus could only be written to the file once fully labeled.

3) **Design of a question classification method**: The result of question classification can make a direct impact on the accuracy of the QA system. A unified standard is absent for Chinese question classification, especially in domain-specific QA systems. Therefore, classification methods need to be designed based on data characteristics.

The home appliance KG was used as a knowledge base. Meanwhile, the entity types with properties were extended to "entity class" and "entity-property class". The entity type is regarded as the entity class, such as home appliance and company. The entity's property belongs to the "entity-property class", and each property is regarded as a subclass, as shown in Table II.

Introducing a subclass can subdivide the question class to improve the accuracy of question classification and further
clarify the question intention. For example, the question "美的 BCD-606WKPZM 的参数有哪些？" belongs to "家电 (home appliance)" without a specific property class, while the question "美的 BCD-606WKPZM 的节能等级是什么？" is directly ascribed to "家电-能效等级 (home appliance-energy efficiency level)". By NER, the latter's semantics was found to ask about the energy efficiency level of the home appliance with a model of "BCD-606WKPZM". Then, a Cypher query statement "MATCH (n)-[r:`] RETURN b" was constructed for this question, and the result was returned to the user after querying the KG to realize the automated QA.

<table>
<thead>
<tr>
<th>Class</th>
<th>Subclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>家电 (home appliance)</td>
<td>/</td>
</tr>
<tr>
<td>家电-属性 (home appliance-property)</td>
<td>家电-能效等级 (home appliance-energy efficiency level)</td>
</tr>
<tr>
<td>品牌 (brand)</td>
<td>/</td>
</tr>
<tr>
<td>品牌-属性 (brand-property)</td>
<td>品牌-成立时间 (brand-establishment date)</td>
</tr>
<tr>
<td>公司 (company)</td>
<td>/</td>
</tr>
<tr>
<td>公司-属性 (company-property)</td>
<td>公司-简介 (company-brief introduction)</td>
</tr>
</tbody>
</table>

C. RoBERTa-WWM Pre-trained Language Model

RoBERTa-WWM pre-trained language model is an improved version of the BERT model [7]. It is more suitable for Chinese NLP tasks due to using large-scale training data, removing the Next Sentence Prediction pre-training task, changing static masking into dynamic masking, and introducing WWM. The node property information of the home appliance Chinese KG is composed of word groups, so the RoBERTa-WWM model is more suitable for home appliance QA than the BERT model. Based on the former, the question corpus dataset was used to fine-tune the model, and question NER and semantic classification were designed and achieved, which could improve the quality of the QA model.

D. NER Model based on RoBERTa-WWM-BiLSTM-CRF

The QA model needs to extract entity information of questions by NER first. With corpus enrichment and computing power improvement, the NER task generally uses the method based on a deep learning model. Due to both close relations, Chinese semantics should be determined by combining the context; therefore, the improved RoBERTa-WWM pre-training language model was used for the Chinese NLP task. Herein, with BiLSTM-CRF [8] as a baseline model, the RoBERTa-WWM was introduced to construct the RoBERTa-WWM-BiLSTM-CRF model (Fig. 3) to carry out the NER of the home appliance questions.

The model mainly comprises RoBERTa-WWM Chinese pre-trained language model, Bi-directional Long Short-Term Memory (BiLSTM) layer, and Conditional Random Field (CRF) layer. RoBERTa-WWM encodes the input text and passes the obtained sequence vector representation to BiLSTM. The BiLSTM layer performs further semantic encoding of the sequence vectors, after which it outputs the score of each class label corresponding to each character. In the CRF layer, the final prediction sequence is obtained by the constraint information between the adjacent labels to get the suitable class of each character. The algorithm module was introduced in the following.

![Fig. 3 Structure of RoBERTa-WWM-BiLSTM-CRF model](image)

1) BiLSTM: The RoBERTa-WWM layer treats the training data to obtain an accurate semantic vector representation. The input of the result into the BiLSTM network can improve the model's ability to capture the dataset's contextual features. BiLSTM consists of LSTMs, of which everyone contains a forget gate, an input gate, and an output gate. These gates control the proportion of information that is forgotten and passed on to the next time node. The calculation formulas can be found in a previous study [9].

2) CRF: The BiLSTM can output meaningless characters without considering the dependency between adjacent labels. As a discriminant model, the CRF adds some practical constraints between labels to reduce the number of invalid predictive label sequences significantly. According to the sequence input, CRF can predict the corresponding state sequence and consider inputting the dependency between the current state feature and each label to find the optimal label sequence. In the CRF model, the output sequence $X = \{x_1, x_2, \ldots, x_n\}$ of the BiLSTM layer corresponds to a set of candidate state sequence labels $Y_s$. The final label sequence $Y = \{y_1, y_2, \ldots, y_v\}$ is determined by calculating the score of each label sequence, as shown in (1).

$$
\text{score}(x, y) = \sum_{i=1}^{n} P_{ij} + \sum_{i=1}^{n} A_{ij}
$$

The symbol $P$ represents the score matrix with a size of $n \times k$ that BiLSTM outputs, where $n$ and $k$ indicate the sequence number of words and tags, and $P_{ij}$ means the score of the $i$-th tag of the $i$-th word. The symbol $A$ indicates the transition score matrix with a size of $k \times 2$ containing the start and end tags of the sentence, and $A_{ij}$ represents the fraction of the process that label $i$ is transferred to label $j$. 

638
Finally, normalizing each label sequence's score can obtain the probability (2). The final tag sequence of the sentence shows the highest probability.

\[
P(y|x) = \frac{\exp(score(x, y))}{\sum_{y'} \exp(score(x, y'))}
\]  

(2)

E. Question Semantic Classification Model based on RoBERTa-WWM-TextCNN

A deep learning model was used to classify questions because the traditional question classification method has low generalization ability and high calculation cost. Compared with the Word2vec word vector, RoBERTa-WWM pre-training model has a higher text representation ability in Chinese NLP tasks; meanwhile, the short text classification model TextCNN outperforms conventional models in terms of simple structure and fast training speed [10]. Therefore, a question semantic classification model RoBERTa-WWM-TextCNN was proposed. First, the question context modeling and sentence-level semantic representation were carried out based on the pre-trained model RoBERTa-WWM. Then, the vector corresponding to the "[CLS]" symbol output from the RoBERTa-WWM last layer was used to represent the question context. Finally, feature extraction and subsequent classification process were carried out by using the fully connected layer of TextCNN. As shown in Fig. 4, the pipeline of the method is composed of a word embedding layer, a convolutional layer, a max-pooling layer, and a fully connected layer.

![Image: Question semantic classification model based on RoBERTa-WWM-TextCNN](image)

| Fig. 4 Question semantic classification model based on RoBERTa-WWM-TextCNN |

a) Word embedding layer: By encoding input text data using RoBERTa-WWM, a word vector matrix \( E \in \mathbb{R}^{n \times d} \) was generated, where \( n \) and \( d \) represent the length of the input sentence and the dimension of the word vector, respectively. For \( x_i \in \mathbb{R}^d \) representing the \( i \)-th input word vector, the input sentence \( X \) can be described as \( X = [x_1, x_2, \ldots, x_n] \).

b) Convolutional layer: The convolution module of TextCNN was employed to extract the intrinsic features of the input text. Notably, the width \( d \) of the convolution kernel \( w \in \mathbb{R}^{h \times d} \) is consistent with the dimension of the word vector. Here, \( h \) represents the height of the convolution kernel, and the widths of three convolution kernels were 3, 4, and 5. The input and output channels of the convolution kernel were determined to be 1 and 256, respectively. The convolution kernel was convolved with the \( i \)-th window \( x_{i:i+1} \in \mathbb{R}^d \) of the word vector matrix \( E \) to obtain the feature \( c_i \) (3). Herein, \( f \) and \( b \) represent the ReLU activation function and the bias, respectively. We used (3) to obtain several features, followed by concatenating to produce a feature vector (4) with a dimension of \( n-h+1 \).

\[
c_i = f(w \cdot x_{i:i+1} + b)
\]  

(3)

\[
c = [c_1, c_2, \ldots, c_{n-h+1}]
\]  

(4)

c) Pooling layer: Each feature was obtained by max-pooling operation (5). Then, the results of all convolution kernels were concatenated to form a new feature vector.

\[
c_e = \max \{c\}
\]  

(5)

d) Fully connected layer: In this layer, dropout was added to prevent overfitting, and the softmax function was used to output the probability of each classification. Finally, the classification result with the highest probability was output.

III. EXPERIMENTS

A. Dataset

The home appliance KG consisted of 27,408 entity nodes and 57,440 relations. The original questions were derived from the relevant data of e-commerce websites or obtained by corpus generation based on rules. The questions of 32,652 obtained by these two methods were divided into a training set, a test set, and a verification set based on a ratio of 8:1:1.

1) NER dataset: The NER dataset was obtained by BIO sequence labeling of the question corpus, of which the major labeled entities contain the model, brand, type, and company of home appliances. The maximum number of models was as high as 13,068, as shown in Table III, and the labeled data were written into files in a format of "<word, label>".

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Label</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home appliance model</td>
<td>MODEL</td>
<td>13068</td>
</tr>
<tr>
<td>Home appliance type</td>
<td>TYPE</td>
<td>23</td>
</tr>
<tr>
<td>Company</td>
<td>COMPANY</td>
<td>158</td>
</tr>
<tr>
<td>Brand</td>
<td>BRAND</td>
<td>188</td>
</tr>
<tr>
<td>Placeholder word</td>
<td>NONE</td>
<td>/</td>
</tr>
</tbody>
</table>

| Table III. Labels of NER data |

2) Dataset of question semantic classification: The dataset was obtained by tagging and classifying the original questions according to the abovementioned classification method. Then, data were written to the files of the training set, the test set, and the validation set in the format of "<question, classification, tag>" as shown in Table IV.
The BiLSTM-CRF model exhibited superior performance to the BiLSTM model, indicating that the result obtained using BiLSTM alone may not be a globally optimal label sequence. CRF module can analyze the relation between adjacent labels to get the optimal entity label, thereby improving the precision of NER. Compared with the BiLSTM-CRF model, the $P$, $R$, and $F_1$ values of the CNN-BiLSTM-CRF model were increased by 2.8%, 1.6%, and 2.2%, respectively, indicating that the CNN module is active in extracting local features. The combination of CNN and global features extracted by BiLSTM stood out in the NER task. The RoBERTa-WWM-BiLSTM-CRF model exhibited the highest $P$, $R$, and $F_1$ values because the RoBERTa-WWM pre-trained language model has an excellent feature extraction ability to express the semantic information of Chinese words better.

The RoBERTa-WWM-BiLSTM-CRF model could recognize four kinds of entities well, while the recognition result of MODEL was lower than those of other entities (Fig. 5). It was due to many letters and combinations in the home appliance model, which hindered the model from learning characteristics.

### Results and Discussion

#### Home appliance NER

The RoBERTa-WWM-BiLSTM-CRF model was compared with several common models (such as BiLSTM, BiLSTM-CRF, and CNN-BiLSTM-CRF [11]) to verify its effect in NER, as shown in Table VII.

### Table IV. Examples of question semantic classification

<table>
<thead>
<tr>
<th>Question</th>
<th>Classification</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>小米扫地机器人STYTJ02ZHM有多重？</td>
<td>家电-商品重量</td>
<td>1</td>
</tr>
<tr>
<td>(What is the weight of the Xiaomi sweeping robot STYTJ02ZHM?)</td>
<td>(home appliance-product weight)</td>
<td></td>
</tr>
<tr>
<td>热水器JSQ25-S6D13的参数有哪些？</td>
<td>家电 (home appliance)</td>
<td>2</td>
</tr>
<tr>
<td>(What are the parameters of the water heater JSQ25-S6D13?)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>美的集团股份有限公司的口号是什么？</td>
<td>公司-口号</td>
<td>3</td>
</tr>
<tr>
<td>(What is the slogan of Midea Group Co., LTD?)</td>
<td>(company-slogan)</td>
<td></td>
</tr>
</tbody>
</table>

### Table V. Parameter setting of NER

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden_layers</td>
<td>12</td>
<td>Droupout</td>
<td>0.5</td>
</tr>
<tr>
<td>Hidden_state</td>
<td>768</td>
<td>Learning_rate</td>
<td>1e-5</td>
</tr>
<tr>
<td>Hidden_unit (LSTM)</td>
<td>128</td>
<td>Max_seq_length</td>
<td>128</td>
</tr>
<tr>
<td>Epoch</td>
<td>20</td>
<td>Batch_size</td>
<td>32</td>
</tr>
</tbody>
</table>

### Table VI. Parameter setting of question semantic classification

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden_layers</td>
<td>12</td>
<td>Droupout</td>
<td>0.2</td>
</tr>
<tr>
<td>Hidden_state</td>
<td>768</td>
<td>Learning_rate</td>
<td>5e-6</td>
</tr>
<tr>
<td>Num_filters</td>
<td>256</td>
<td>Max_seq_length</td>
<td>128</td>
</tr>
<tr>
<td>Kernel_size</td>
<td>3, 4, 5</td>
<td>Epoch</td>
<td>10</td>
</tr>
</tbody>
</table>

### Evaluation Index

The values of precision rate ($P$), recall rate ($R$), and $F_1$, calculated by (6–8), were used as evaluation indexes in NER and question semantic classification. The symbols $T_p$, $F_p$, and $F_N$ represent the number of positive samples also predicted to be positive, the number of negative samples predicted to be negative, respectively. Therefore, $T_p + F_p$ indicates the total number of samples predicted to be positive, while $T_p + F_N$ corresponds to the total number of positive examples in the dataset. The harmonic mean of $P$ and $R$ is represented by $F_1$, which can provide a more comprehensive evaluation of the model. A higher $F_1$ value indicates a more effective model.

$$P = \frac{T_p}{T_p + F_p}$$  \hspace{1cm} (6)

$$R = \frac{T_p}{T_p + F_N}$$  \hspace{1cm} (7)

$$F_1 = \frac{2PR}{P + R}$$  \hspace{1cm} (8)

#### Results and Discussion

1) **Home appliance NER**: The RoBERTa-WWM-BiLSTM-CRF model was compared with several common models (such as **BiLSTM, BiLSTM-CRF, and CNN-BiLSTM-CRF** [11]) to verify its effect in NER, as shown in Table VII.

### Table VII. NER results of different models

<table>
<thead>
<tr>
<th>Model</th>
<th>$P$ (%)</th>
<th>$R$ (%)</th>
<th>$F_1$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiLSTM</td>
<td>85.6</td>
<td>82.8</td>
<td>84.1</td>
</tr>
<tr>
<td>BiLSTM-CRF</td>
<td>89.7</td>
<td>88.5</td>
<td>89.1</td>
</tr>
<tr>
<td>CNN-BiLSTM-CRF</td>
<td>92.5</td>
<td>90.1</td>
<td>91.3</td>
</tr>
<tr>
<td>RoBERTa-WWM-BiLSTM-CRF</td>
<td>94.6</td>
<td>93.5</td>
<td>94.0</td>
</tr>
</tbody>
</table>

The $P$ of the three models reached more than 90%, indicating that the deep learning model performed well in question semantic classification. TextCNN and BiLSTM showed similar results, and the $P$ of the former was only 0.8%.
higher, while it exhibited a simpler model and faster training speed. The introduction of TextCNN into the RoBERTa-WWM pre-trained language model increased the values of all indicators, which were higher than those of the other two models. For example, compared with the TextCNN model, the $P$, $R$, and $F1$ values of the RoBERTa-WWM-TextCNN model were improved by 2.0%, 2.4%, and 2.2%, respectively. It was ascribed to a multi-headed attention mechanism introduced by the RoBERTa-WWM model, which focused on the word in a sentence corresponding to a specific word from multiple perspectives. In contrast, the TextCNN model was poor in extracting semantic features because it was limited by the length and number of CNN convolution kernels. The RoBERTa-WWM model could learn word embedding features with contextual semantics using an unsupervised method in the pre-training stage to absorb much complex linguistic knowledge, which could express semantics better. The model could also adapt to the relevant contextual information in the subsequent fine-tuning stage, thereby working better than the abovementioned models.

E. Automated QA

The model, generated from NER and question semantic classification, can be used for automated QA. The algorithm models can recognize the entity and classification information of the question. The latter corresponds to the relation in the KG, which can convert the question into triplet information <entity, property,?> that the Neo4j graph database can understand, "?" representing the answer the user wants.

The question "小米扫地机器人 STYTJ02ZHM 有多重? (What is the weight of the Xiaomi sweeping robot STYTJ02ZHM?)" was taken as an example (Table IX). Model parsing was used to determine its entity and semantic classification information. Then, a Cypher query statement was constructed based on the abovementioned information, followed by querying the answer in the KG. The system would conduct a fuzzy query if the entity information were incomplete, e.g., "小米扫地机器人 STYTJ0 有多重? (What is the weight of the Xiaomi sweeping robot STYTJ0?)". The results were filtered by combining other entity information (such as "小米 (Xiaomi)" and "扫地机器人 (sweeping robot)") of the question. The remaining results were then matched with the question by similarity, returning the answer with the highest matching degree. If no entity or classification information were obtained after model parsing, the system automatically would return a friendly prompt.

<table>
<thead>
<tr>
<th>TABLE IX. EXAMPLES OF AUTOMATED QA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the question: 小米扫地机器人 STYTJ02ZHM 有多重?</td>
</tr>
<tr>
<td>NER:</td>
</tr>
<tr>
<td>BRAND: [小米]</td>
</tr>
<tr>
<td>TYPE: [扫地机器人]</td>
</tr>
<tr>
<td>MODEL: [STYTJ02ZHM]</td>
</tr>
<tr>
<td>Semantic classification: 家电-商品重量</td>
</tr>
<tr>
<td>Construct the Cypher query: MATCH (n)-[r:<code>商品重量</code>]-&gt;(b) where n.name=&quot;STYTJ02ZHM&quot; RETURN n,b,r.</td>
</tr>
<tr>
<td>Return the answer: STYTJ02ZHM 的商品重量为 4.7 kg.</td>
</tr>
</tbody>
</table>

IV. CONCLUSIONS

An intelligent question answering (QA) model based on RoBERTa-WWM was developed according to the characteristics of Chinese questions in home appliances. Based on the data characteristics of the home appliance knowledge graph (KG), the question corpus generation method was designed, and the corpus dataset was constructed. Then, the named entity recognition model (RoBERTa-WWM-BiLSTM-CRF) and the question semantic classification model (RoBERTa-WWM-TextCNN) were constructed to parse the question semantics. Finally, the model’s effectiveness was verified by comparing it with other deep learning models, and the function of automated QA was completed based on the training model.

In the future, we will work on acquiring more relevant knowledge in question corpus generation. In addition, more methods of multi-round continuous QA based on KG need to be developed under Chinese semantics.

REFERENCES

Variational Autoencoder with Stochastic Masks to Solve Exposure Bias in Recommendation

Yingshuai Kou∗†‡, Neng Gao∗‡, Yifei Zhang∗‡
∗State Key of Laboratory of Information Security, Chinese Academy of Sciences, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
‡Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{kouyingshuai, gaoneng, }@iie.ac.cn

Abstract—In the recommended scenario, an unobserved interaction may exist in two cases: the user is not interested in the item or the user is not aware of the item at all. This phenomenon leads to a serious exposure bias problem in recommendation system. To solve this problem, we propose Masked Variational Autoencoder (MVAE). Firstly, we predict the missing values in the sparse user-item interaction matrix by matrix completion. Then we randomly mask the elements in the obtained matrix and use them as the input to the variational autoencoder. The decoder can reconstruct the user interaction matrix closer to the true distribution and fully exploit the potential preferences of users in uninteracted items. In particular, we use a combined dual VAE to tackle the exposure bias problem from the user side and the item side respectively. Extensive experiments on three real-world datasets also illustrate the effectiveness of MVAE for solving exposure bias in recommendation.

Index Terms—Recommendation System, Exposure Bias, Variational Autoencoder

I. INTRODUCTION

Recommendation system is a subfield of software engineering that uses machine learning, data mining and other technologies to actively show users items they might like during their browsing interactions[1]. However, user interaction data is observation-based, the full picture of the data is unknown. This leads to a serious exposure bias problem. In the actual scenario, the unobserved sample may exist in two cases, i.e., the item does not match the user’s interest or the user doesn’t know the item at all. After clarifying the causes of exposure bias, we need to figure out why the existing methods do not work well to solve the exposure bias problem.

a) Deficiencies in model evaluation: Most Recommendation methods [5] is optimized by randomly sampling among uninteracted items as negative examples and maximizing the distance between positive and negative examples. However, as mentioned above, this optimization approach undoubtedly exacerbates the exposure bias.

b) Models lack the ability to generate: Traditional self-supervised approach such as Autoencoder model[12] reconstructs the data by encoding and decoding operations. But the users and items are directly mapped into hidden vectors, the models can’t explore the hidden potential preferences of uninteracted users sufficiently.

c) Unable to handle missing values in sparse data: Exposure bias is mainly due to sparse data. Even the generative model is still insufficient to mine users’ potential interest preferences from a large number of uninteracted samples.

Hence, we propose the Masked Variational Autoencoder (MVAE) method to solve the exposure bias problem in the recommendation. The origin interaction matrix is first complemented by the Singular Value Decomposition(SVD). SVD maps the users and items into a same vector space and complements the missing values in the matrix by constructing links between user items. We mask a random part of pseudo-matrix and reconstruct the missing elements by VAE. With the generalization capability of masking and the generation capability of VAE, we expect to be able to restore the interaction matrix which is closer to the true distribution. In particular, we design a combined dual VAE structure that combines the exposure of items to the user on the user side and the display of items from the item side.

In summary, our contributions in this paper are as follows:

- To solve the problem that the uninteracted samples contain a large number of potential user preferences due to data sparsity, we use the SVD method to complete the matrix, and the obtained matrix is randomly masked to be the input of VAE.
- We propose a combined dual VAE structure, the user side is used to predict users’ potential items of interest, while the item side is used to expose items to interested users, the two-way design effectively increases the robustness of the model.
- We conduct experiments on three real-world recommendation scenarios, and the experimental results show that our method has a better effect of exposure debiasing.

II. RELATED WORK

Research methods for exposure bias can be divided into two categories: debiasing in evaluation and debiasing in training.

The main approach to eliminate exposure bias in the evaluation phase is to use the propensity score. For example, SNIPS [11] evaluates the error of implicit feedback data on traditional metrics such as AUC, DCG@k, Recall@k, and later uses an inverse propensity score framework to offset the exposure bias. The current mainstream approaches try to solve this problem during the training phase of the model. One of them
is based on heuristic strategies. For example, user-item feature similarity [6] has also been used to define the confidence level. Another type of research route is resampling, some researchers have explored the use of auxiliary information to augment the sampler. SamWalker++ [9] adds social networks as auxiliary information to the sampling distribution of the model. Although such methods achieve some debiasing effect, the addition of auxiliary information increases the data processing load and the computational complexity of the model.

III. Problem Formulation

We start by characterizing the recommendation problem and introducing notation to be used throughout. In a classic implicit recommendation scenario, we denote the set of users as \( U = \{ u_1, u_2, \ldots, u_n \} \), and the set of items as \( I = \{ i_1, i_2, \ldots, i_m \} \). The history feedback from users to items can be represented as \( O = \{ u, R_{ui} \} \), where \( R_{ui} \) indexes the rating from the user to the item \( i \). Based on this interaction set \( O \), we can construct an user-item interaction matrix \( R \in \{0, 1\}^{n \times m} \), where \( R_{ui} = 1 \) if the interaction of a user \( u \) and an item \( i \) is observed. The rest of the unobserved samples are denoted by 0 as \( R_{ui} = 0 \). The intention of the recommendation system is to predict whether a user is interested in a candidate item that he has not interacted with before. Our goal is to learn a prediction function to calculate the probability that user \( u \) will click item \( i \).

IV. Methodology

In this section, we illustrate the proposed MVAE in detail. The overall architecture of the model is shown in Fig. 1.

A. Matrix Completion

First of all, we use matrix factorization to complete the matrix. That is, we fill the elements of the matrix that have not generated records (unobserved data) based on the existing data in the matrix (observed data). Here we choose Singular Value Decomposition (SVD) as the matrix completion method. Interaction matrix \( R \) can be linearly combined by two smaller matrices according to the SVD principle.

\[
R = PQ^T \tag{1}
\]

where the original interaction matrix is decomposed into users features matrix \( P \) and item features Matrix \( Q \). In the decomposed matrix, each user and item is represented as a feature vector consisting of \( K \) features \( f \). The \( P \) and \( Q \) matrices are obtained by training in the learning process, and a loss function is defined as:

\[
E = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij} (R_{ij} - p(P_i, Q_j))^2 + \frac{k_1}{2} \sum_{i=1}^{n} \|P_i\|^2 + \frac{k_2}{2} \sum_{j=1}^{m} \|Q_j\|^2 \tag{2}
\]

where \( p(P_i, Q_j) \) represents the prediction score of user \( i \) and item \( j \). The prediction function \( p \) is the dot product \( p(P_i, Q_j) = P_i Q_j^T \). \( I \in \{0, 1\}^{n \times m} \) in Eq. (3) is an indicator of whether the corresponding position has a interaction. The last two terms on the right side of the equation are the regularization terms to prevent overfitting. After training, we can complete the missing values in the original matrix by the product of matrices \( P \) and \( Q \). However, the value of the complement here is based on predictions, which is still not the exact true value. We temporarily call it the pseudo-matrix \( R' \).

B. Stochastic Masks

We mask the pseudo-matrix \( R' \) in order to further increase the generalization capability of the model. To be specific, we divide the matrix into submatrices at random in a certain ratio. Then we sample one of the submatrices and mask the remaining submatrices. Here we directly set the values of these positions to 0. Our sampling strategy is simple: we randomly sample the elements of the matrix without replacement according to a uniform distribution. Random sampling on the one hand largely eliminates redundancy, and on the other hand guides the model in learning which values are true in our pseudo-matrix \( R' \).

C. Variational Autoencoder

The variational autoencoder (VAE) reconstructs the matrix by mapping the original data into a standard normal distribution and desampling a representation from the distribution for decoding. For ease of exposition, the input is uniformly denoted by \( x \). In terms of a probabilistic model, the encoder can also be defined by the parametrized posterior distribution \( p(z|x) \). Here we use the SVD to complement the original matrix and mask some of the values to obtain a new \( \tilde{R} \). Then the encoder fits \( p(z|\tilde{R}) \) based on \( \tilde{R} \). The value of the matrix complement guides us to discover which of the unobserved samples are potentially positive, and by assigning values to them we obtain an \( \tilde{R} \) distribution that is closer to the true distribution. However, since the posterior probability distribution is difficult to solve in probabilistic models, for each data point we need to approximate the intractable posterior distribution \( p(z|\tilde{R}) \). Therefore, we need to use variational inference [3]. Variational inference maximizes the approximation of the original posterior distribution by a simple variational distribution \( q(z) \). We set \( q(z) \) to be a fully factorized (diagonal) Gaussian distribution:

\[
q(z) = \mathcal{N}(\mu, \text{diag} \{ \sigma^2 \}) \tag{3}
\]
Then the distribution is parametrized by \( \theta \) with both multivariate functions \( \mu_{\theta}(\mathbf{x}) \) and \( \sigma^2_{\theta}(\mathbf{x}) \) being K-vectors and sets the variational distribution as follows:

\[
q_{\theta}(z, \mathbf{x}) = \mathcal{N}(\mu_{\theta}(\mathbf{x}), \sigma^2_{\theta}(\mathbf{x}))
\] (4)

The encoder of VAE uses the masked \( \mathbf{x} \) as input and derives the corresponding variational parameters of the variational distribution \( q_{\theta}(z, \mathbf{x}) \) by the inference model.

VAE’s decoder model can be viewed as a generative model \( p_{\xi}(x | z) \), sampling \( z \) from the distribution fitted by the encoder and then reconstructing the original vector \( x \). According to the method of learning latent variable models by variational inference, we can calculate the Evidence Lower Bound (EBLO) of the data, that is, the final derivation in Eq. (5). We use it as the objective to seek maximization of the VAE reconstruction results.

\[
\log p(x; \xi) \geq E_{q_{\theta}(z|x)} [\log p_{\xi}(x|z)] - KL (q_{\theta}(z) \| p(z))
\] (5)

Then we use the reparametrization trick by introducing the regularization hyperparameter \( \beta \) to control the trade-off between the regularization term (i.e., KL loss) and the reconstruction loss. Thus the loss function of the VAE is as follows:

\[
L_{\beta}(x; \xi, \theta) = E_{q_{\theta}(z|x)} [\log p_{\xi}(x | z)] - \beta \cdot KL (q_{\theta}(z | \mathbf{x}) \| p(z))
\] (6)

VAE makes flexible use of variational inference, and generates data to answer relevant new questions. The ability to generate inference in this way can help us to better address the problem of exposure bias by exploring the missing values in the interaction matrix together with the mask operation.

D. Combined uVAE and iVAE

We construct a combined model of user-based VAE(uVAE) and item-based VAE(iVAE). The joint optimization of these two VAEs contributes to their fine-tuned calibration, and together uVAE and iVAE can learn complementary information from the user’s interaction with the item. For the user-item interaction matrix \( \mathbf{R} \), the uVAE reconstructs the matrix row-by-row, while the item VAE reconstructs it column-by-column.

The final predicted output:

\[
\hat{\mathbf{R}} = \alpha \hat{\mathbf{R}}_u + (1 - \alpha) \hat{\mathbf{R}}_i
\] (7)

where \( \hat{\mathbf{R}}_u \) and \( \hat{\mathbf{R}}_i \) are the uVAEs’ and iVAEs’ output reconstruction matrices, respectively. Here we follow the experimental setup of the joint method described above, taking \( \alpha = 0.5 \). According to Eq. (9), we add the loss functions of uVAE and iVAE as the total loss function of our MVAE model:

\[
L_{\text{MVAE}}(\mathbf{R} | \mathbf{x}, \beta) = \sum_{u \in U} L_{\text{VAE}}(\mathbf{R}_u |, \mathbf{x}_u, \beta) + \sum_{i \in I} L_{\text{VAE}}(\mathbf{R}_i^T |, \mathbf{x}_i, \beta)
\] (8)

where \( \mathbf{x}_u \) and \( \mathbf{x}_i \) represent the model parameters of uVAE and iVAE, respectively. It should be noted that, unlike the traditional implicit feedback recommendation algorithms, we do not use the pairwise BPR ranking loss which is optimized by maximizing the distance between positive and negative examples. As mentioned above, the negative cases randomly sampled from the uninteracted samples are not necessarily negative, so the loss will inadvertently increase the exposure bias. We use the self-supervised VAE model and optimize the reconstruction loss in a way that can effectively avoid this phenomenon.

V. EXPERIMENTS

In this section, we evaluate our proposed model MVAE and present its performance on three real-world datasets.

<table>
<thead>
<tr>
<th></th>
<th>Movielens1M</th>
<th>Yelp</th>
<th>Pinterest</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Users</td>
<td>6,027</td>
<td>12,705</td>
<td>55,187</td>
</tr>
<tr>
<td>#Items</td>
<td>3,062</td>
<td>9,245</td>
<td>9,911</td>
</tr>
<tr>
<td>#Interaction</td>
<td>574,026</td>
<td>318,314</td>
<td>1,500,806</td>
</tr>
<tr>
<td>#Sparsity</td>
<td>96.89%</td>
<td>99.73%</td>
<td>99.73%</td>
</tr>
</tbody>
</table>

A. Datasets

To evaluate the effectiveness of our model, We conduct experiments based on three real-world datasets. Basic statistics of the datasets are summarized in Table I. Movielens1M is a widely used benchmark dataset in movie recommendations. Yelp is a subset of Yelp’s businesses, Pinterest is an image based content social networking site. Following the previous work [2], [13], we keep only users and items with at least 20 interactions to single out good quality data from the original dataset by using above setting.

B. Baselines


C. Evaluation Protocols and Parameters

We utilize two commonly-used metrics to assess the quality of predicted ranked list for each user: \( F1\text{-Score}@K \) and \( NDCCG@K \). We report the average of these metrics (over testing users). We optimize our model with Adam, and set the learning rate to 0.003. For the training data of each batch, we decompose the matrix into 1500 \( \times \) 1500 submatrices. The hyperparameters were set as: \( \beta = 0.15, \alpha = 0.5 \). The mask ratio is 4% for each epoch. For each encoder and decoder in the VAE, we have two hidden layers, each of dimension size
320, and the activation function used is tanh. The dimension of the potential space $d$ is set to 80. The final output layer uses the sigmoid activation function. We train all the models on a single NVIDIA Tesla A100 GPU for 200 epochs.

**D. Overall Performance**

Table II reports F1-score and NDCG for all datasets and methods. Each metric is averaged across all test users. MVAE significantly outperforms the baselines across datasets and metrics. On the MovieLens1M dataset, we have made some significant improvements over the optimal baseline algorithm JoV, especially in the NDCG@K evaluation metric. This is mainly because MVAE takes the masked complementary matrix as input and assigns different propensity scores to those potential points of interest. The improvement on the Pinterest dataset was significant, with the best metric improving by 9.375% (NDCG@10). The minimum improvement is 5.769% (NDCG@5). However, we also note that the result of $K = 1$ metric is sub-optimal on the Yelp dataset. Probably because the number of items in Yelp is too large and it is not as efficient when exposing unseen items. But as $K$ increases, our effect improves and becomes optimal. This also suggests that our model gradually increases the exposure of items, which will further improve the recommendation results as the feedback loop of the recommendation system progresses. Overall, these results indicate that MVAE achieves better exposure debiasing results than traditional methods and significantly improves the accuracy of the recommendations.

**E. Study on MVAE Structure**

![Fig. 2. Test performance of uV AE, iV AE and MV AE in three datasets.](Image)

In this section, we conduct ablation studies on uV AE and iV AE separately to demonstrate that our combined model is effective. Fig. 2 illustrates the results. From the experimental results, it can be seen that MAVE improves significantly over both uV AE and iV AE by coupling the dual VAEs from the user side and the item side. It indicates that uV AE and iV AE generate different recommendations which are complementary to each other.

**VI. CONCLUSION**

In this work, we propose a dual Variational Autoencoder with stochastic masks to solve exposure bias in recommendation. Experiments on three real world datasets confirm the effectiveness of our model.

**REFERENCES**


Anaphora Ambiguity Detection Method Based on Cross-domain Pronoun Substitution

Fengyong Peng†, Xi Wu†, Yongxin Zhao*†§, and Yongjian Li§
† Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
‡ The University of Sydney, Australia
§ State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China

Abstract—Pronoun anaphora ambiguity is very common in natural language descriptions, especially in specialized fields such as computing, medicine and aerospace. When multiple antecedents appear before a pronoun word, readers with different background knowledge often have completely different understandings on a same word. In order to reduce such misunderstandings caused by ambiguity in the process of document propagation, we usually use manual methods to check the ambiguity of reference, which however cannot meet the increasing needs of detection with the development of various disciplines. In this paper, we propose a method to intelligently detect sentences with anaphora ambiguity. First of all, we identify criteria for ambiguous sentences and then use word embeddings to further detect ambiguity. Specifically, we propose a pronoun substitution strategy based on coreference resolution, and combine this strategy with word embedding techniques to generate a cross-domain anaphora ambiguity detection method. Finally, we carry out experiments on aerospace documents, which verify the effectiveness of our proposed method in anaphora ambiguity detection.

Index Terms—ambiguity detection, anaphora ambiguity, pronoun substitution, Cross-domain, natural language processing

I. INTRODUCTION

There are a large number of documents closely related to professional background knowledge in various subject fields. In the process of document transmission, there are usually differences in understanding due to the different knowledge background of readers. We call such differences in understanding cross-domain ambiguity. The cost of manual ambiguity detection is getting higher and higher, and it has become an urgent problem to use intelligent technology to detect ambiguity in documents in specific fields.

In the field of natural language processing, the purpose of ambiguity detection is to first set an ambiguity standard and then judge whether a sentence can be interpreted in multiple ways under this standard. Ferrari et al. have successively proposed the domain-specific ambiguity detection method based on word embedding [1], the cross-domain ambiguity identification method based on language model [2], and the cross-domain ambiguity detection method oriented to requirements engineering [3]. On this basis, Vaibhav Jain et al. proposed cross-domain ambiguity detection using linear transformation of word embedding spaces [4], the word vector model is trained on the corpus of different fields, and then the word vectors obtained from different fields are mapped to the same space by using linear transformations. The ambiguity of words is measured by the difference of word vectors. Siba Mishra et al. [5] have shown, through extensive and detailed experiments on several different subdomains, that word embedding based techniques are very effective in identifying domain-specific ambiguous words. The above studies have achieved good results in cross-domain ambiguity detection, but the above schemes also have some shortcomings. They only solve the problem of cross-domain ambiguous words without exploring the ambiguity of sentences in depth.

Inspired by the above work, we propose anaphora ambiguity detection method based on cross-domain pronoun substitution. The proposed method is tested in the aerospace field. Experimental results verify the effectiveness of the proposed method in anaphora ambiguity detection. Our major contributions include the following two points:

1) Our method combines pronoun substitution with cross-domain ambiguity detection and achieves good results in anaphora ambiguity detection.
2) The cross-domain ambiguity detection method based on word embedding achieves good results at the word level. We consider the weight of words, design the construction method of sentence embedding, and realize the ambiguity detection at the sentence level.

The rest of this paper is structured as follows: Section II discusses related work; Section III describes the research methods; Section IV shows the result through experiment; Section V concludes our work and looks forward to future works.

II. RELATED WORK

The ambiguity of pronoun anaphora is a typical kind of ambiguity in natural language processing. We assume that there are multiple alternative antecedents before the pronoun, so we can think that the sentence will produce anaphora ambiguity when there is little difference between the probability of multiple alternative antecedents and the pronoun association. Based on the above assumptions, we use the clustering results obtained by coreference resolution to construct new sentences using the strategy of pronoun substitution, and detect sentence ambiguity by quantifying the difference of sentences after cross-domain pronoun substitution. The construction of the ambiguity detection method involves the following:

*Corresponding author: yzxzhao@sei.ecnu.edu.cn
DOI reference number: 10.18293/SEKE2023-173
A. Coreference resolution

In 2017, Kenton Lee et al. proposed end-to-end neural coreference resolution [6]. Subsequently, Kenton Lee et al. introduced a fully differentiable approximation of higher-order reasoning for coreference resolution [7]. In 2019, Mandar Joshi et al. proposed to apply BERT to coreference resolution [8]. Then Mandar Joshi et al. proposed that improving pre-training by representing and predicting spans model proposed [9] has achieved better coreference resolution results. The technology of coreference resolution is quite mature at present, and our research is carried out on this basis.

B. Embedding spaces alignment

Linear transformations can be used to learn linear mappings from one vector space to another. Mikolov et al. [10] used a set of word pairs \( \{x_i, y_i\}_{i=1}^n \), in which \( y_i \) is the translation of \( x_i \). Then they learn the translation matrix \( W \) by minimizing the following loss function:

\[
\sum_{i=1}^{n} |x_i W - y_i|
\]  

(1)

This method can also be used to align monolingual word embeds. If the meaning of most words is assumed to remain the same, linear regression can be used to find the best rotation alignment between two words embedding spaces. This can be used to identify changes in meaning if words are not properly aligned. This is the basis of the proposed method for identifying cross-domain ambiguous words.

C. Text similarity

Sentence similarity is a complex phenomenon. The meaning of a sentence depends not only on the words in the sentence, but also on the way they are combined. There are also many dimensions of semantic similarity, and sentences may be similar in one dimension and different in others. Smooth Inverse Frequency (SIF) method is a reasonable method for sentence similarity calculation. The SIF method introduces a weight mechanism to reduce the weight of some unimportant words, while retaining the information that contributes more to the semantics. Our paper researches on this basis.

III. ANAPHORA AMBIGUITY DETECTION

The anaphora ambiguity detection method is shown in Figure 1, which mainly includes the following three parts:

A. Construct the coreference resolution model with different knowledge background

To judge whether a sentence is ambiguous, the first problem to solve is to check whether there are ambiguous concepts or polysemous words in the sentence. The space field is very broad, and the background knowledge of the participants is also very different. If the description of fuzzy concept or polysemous word appears in aerospace software carrier, people with different background knowledge are likely to have different understanding. We call such ambiguity phenomenon cross-domain ambiguity, that is, the ambiguity phenomenon caused by the different background knowledge of readers.

Vaibhav Jain et al. [4] showed that word ambiguities can be found by comparing word embeddings across domains. After being applied to the field of coreference resolution, Bert has achieved very good results in coreference resolution. Based on this, we propose to use different background knowledge to construct coreference resolution models. The improving pre-training by representing and predicting spans model proposed by Mandar Joshi et al. [9] has achieved the best results in practical applications. On this basis, we use the corpus from different fields to perform fine-turn operation on the bert pre-trained model to obtain coreference resolution models with different knowledge backgrounds.

B. A pronoun substitution strategy based on maximizing antecedents

Ambiguities caused by unclear references are common in the space industry. Coreference resolution can effectively solve the problem of unclear anaphora in text, but as a clustering operation, coreference resolution cannot be directly used for ambiguity detection. Here, we propose the strategy of pronoun
substitution based on coreference resolution, that is, all the pronouns in the sentence are replaced with antecedents to get a new sentence without pronouns.

We assume that there exists a Chinese sentence $S$, $S = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$. Where $w_5$ is a pronoun, $w_2$ and $w_4$ are nouns, and $w_5$ refers to $w_2$. We assume that $w_2$ is the phrase that can be segmented. So $w_5$ could point to $w_2,w_2-1,w_2-2$, or $w_4$. If we apply the maximization antecedent algorithm 1, then $w_5$ cannot point to $w_2-1$ and $w_2-2$.

Algorithm 1 pronoun replacement algorithm based on maximized antecedents

Input: Original sentence $sen$. Sentence segmentation results(full mode) $words = \{w_1, w_2, \ldots, w_n\}$. Coreference resolution result $clusters = \{c_1, c_2, \ldots, c_m\}$ and $pos = \{p_1, p_2, \ldots, p_m\}$.

Output: The new sentence after the substitution.

1: set $max_w = ''$
2: for $i = 1 \rightarrow n$ do
3: if $(c_i \in w_i \&\& w_i$ is noun $\&\& length(w_i) > length(max_w))$ then
4: $max_w = w_i$
5: end if
6: end for
7: $newSen = sen$
8: for $i = m \rightarrow 2$ do
9: replace $newSen[p_i, p_i + length(c_i)]$ with $max_w$
10: end for
11: return $newSen$

C. Cross-domain ambiguity calculation method based on SIF

In this paper, ambiguity detection is carried out from the sentence level. Firstly, whether there is an anaphora in the sentence is judged. If there is a pronoun case, the pronoun substitution is carried out, otherwise no processing is done. During pronoun substitution, we obtained substitution sentences produced by models with different knowledge backgrounds. Then word Embedding is done on sentences with different knowledge backgrounds. Then, linear variation is used to bring the word embedding models obtained from different domains into a unified word embedding space. In a unified space, we detect ambiguity by analyzing the semantic similarity of two sentences. We believe that there is no ambiguity in the case of basically the same semantics, and there is ambiguity in the case of large semantic differences. Considering that each word in the sentence has different weight for the meaning of the sentence, we use the SIF method to calculate the sentence similarity. Although the traditional SIF method considers the influence of word weight on sentence meaning, it will bring new problems after the introduction of pronoun substitution. We assume that $n$ pronouns have been replaced in the sentence, and the size of $n$ will change the weight of the antecedent in the sentence. In order to eliminate the weight change caused by pronoun substitution errors, we adjust the sentence vector of SIF algorithm and obtain the improved semantic similarity calculation method:

$$vector(sen) = \sum_{w \in sen} weight(w) \cdot IDF_D(w) \cdot vector(w)$$  \hspace{1cm} (2)$$

$$weight(w) = \begin{cases} \frac{1}{|sen|}, & \text{w for original word.} \\ \frac{1}{num|sen|}, & \text{w for substituted word.} \end{cases}$$  \hspace{1cm} (3)$$

We assume that there is a threshold value $\alpha$, when $Simc(sen_1, sen_2) > \alpha$, the sentence is considered ambiguous. When $Simc(sen_1, sen_2) \leq \alpha$, sentence semantic similarity is considered to have no ambiguity.

IV. EXPERIMENT AND ANALYSIS

A. Data

We use web crawler to obtain Chinese space corpus data from Wikipedia and clean the data. In total, we obtained 20,000 articles, 9,132,781 sentences, and 159,288 words.

B. Parameter setting

Coreference resolution We extend the original tensorflow implementation of c2f-coref and bert using the approach of Joshi et al.[9] According to the experimental results of joshi et al., Bert-large model was adopted, and set the $\max_{segment_len} = 384$.

Word Embedding We use the gensim implementation of the word2vec SGNS algorithm for word embedding training. These hyperparameters are the same as those used by Jain et al. [4] in ambiguity detection.

C. Experimental results

Ambiguity calculation results We select 20000 statements from the aerospace software carrier to test the algorithm. In the first step, the corpus is input into the coreference resolution model with different background knowledge, and the anaphora chain and the corresponding position information are output if the sentence has an anaphora. In the second step, based on the original sentence and the anaphora chain, the new sentence substitution operation is carried out by maximizing the antecedent strategy. In the third step, the new sentences obtained in the second step are vectorized under word embedding models with different backgrounds, and then the word embeddings from different spaces are mapped into the same space using linear spatial variation. In the fourth step, in the same word embedding space, the word vectors are accumulated according to the designed weights (formula) to obtain the sentence vectors, and then the cosine similarity of the two sentence vectors is calculated. Finally, the calculation results of ambiguity are shown in the Table I.

Human evaluation results We randomly selected 2000 statements from the 20000 statements that participated in ambiguity computation for manual evaluation. We select two groups of personnel with different background knowledge, one group is the technical personnel in the field of aerospace, and the other group is the administrative staff with no technical background in the field of aerospace. On this basis, the two
groups respectively performed pronoun substitution on 2000 statements according to the pronoun substitution strategy of maximizing the antecedent. Then, the sentence is marked by comparing the results of the two substitutions. If the results of the two substitutions are consistent, it can be considered that the sentence does not have the ambiguity caused by the ambiguity of anaphora. The manual evaluation found anaphora in 447 statements, of which 42 statements had cross-domain ambiguity. We correlate the human standard results and the results of the automated algorithm for detection as shown in the table I.

D. Discussion of ambiguity threshold

We assume that there is an ambiguity threshold $\alpha$, and when the result of the sentence ambiguity calculation is greater than $\alpha$, we can consider the sentence to be ambiguity free. When the result of the sentence ambiguity calculation is less than or equal to $\alpha$, we consider the sentence to be ambiguous. We propose two measures in ambiguity threshold detection. Accuracy of ambiguity detection and coverage of ambiguous sentences. We use the automated ambiguity detection results and the human evaluation results to plot the ambiguity threshold versus the ambiguity detection accuracy and the ambiguity sentence coverage.

By analyzing the images, we think that it is better to set the ambiguity threshold at least 0.7, at this time, it can almost cover most of the ambiguous sentences, and the accuracy of ambiguity detection can also be guaranteed. When the threshold $\alpha$ is 0.7, the ambiguity detection accuracy of the algorithm is 67.11%, and the ambiguity sentence coverage rate is 92.86%.

V. CONCLUSION AND FUTURE WORK

This paper proposes an intelligent method for anaphora ambiguity detection. Based on the research of cross-domain ambiguity and coreference resolution, we propose a cross-domain anaphora ambiguity detection method based on cross-domain pronoun substitution. Experiments in the space field verify the effectiveness of the method.

Domain-specific ambiguity detection is a very complex project. Subsequent work will continue to summarize the rules of other types of ambiguity sentences and study ambiguity detection methods with a wider scope of application.

VI. ACKNOWLEDGEMENT

This work is supported by National Key Research and Development Program (2020AAA0107800), National Natural Science Foundation of China (NSFC 62272165), the “Digital Silk Road” Shanghai International Joint Lab of Trustworthy Intelligent Software (Grant No. 22510750100), and Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

REFERENCES

Multi-source Machine Reading Comprehension with Meta-Learning and Adaptive Adversarial Training

Zhe Wen*, Yatao Qi*, Huan Liu*, Zheng Lin*, Weiping Wang*
*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{wenzhe, qiyatao, liuhuan, linzheng, wangweiping}@iie.ac.cn

Abstract—Machine reading comprehension (MRC) methods have shown great success in many datasets, but existing methods fail to achieve satisfactory results in low-resource scenarios. In addition, existing MRC models suffer from a notable decrease in performance when confronted with scenes different from the training data. Thus, it is hard to transfer knowledge between domains. In this paper, we propose an adaptive meta-learning framework to learn and transfer the shared knowledge. The framework is based on model-agnostic meta-learning algorithm, aiming to aggregate meta-knowledge among multi-source datasets. Furthermore, for better adaptation to different target domain, we investigate an adaptive adversarial training strategy to obtain domain-specific meta-knowledge. We empirically adopt three large-size datasets as source domains and five small-size datasets as target domains, and extensive experiments show the effectiveness of our framework.

Index Terms—Natural Language Understanding, Machine Reading Comprehension, Meta Learning, Adversarial Training.

I. INTRODUCTION

Machine reading comprehension (MRC) attempts to answer many kinds of questions, including simple questions and multi-hop reasoning questions, after reading a document or multiple documents. Recently, MRC models based on the manner of fine-tuning the pre-trained language model for downstream tasks, have achieved acceptable performance [1]. However, these models still rely on an adequate scale of the training data in the target domain. When confronted with the low-resource MRC task, even the performance of models based on pre-training deteriorates significantly.

As shown in Figure 1 as the scale of the labeled data reduced, the performance of BERT-based MRC model decreases by a large margin. Specifically, when approximately 40% data is used for training, the performance curve falls sharply. In fact, for some domains, the lack of adequate labeled data makes it difficult to train a domain-specific MRC model with satisfactory performance. Therefore, it is of great importance and necessity to solve the low-resource problem in MRC.

An intuitive solution is to generalize or transfer knowledge learned in resource-rich domains to resource-poor domains. [2] trained models on one or more source MRC datasets, and then evaluated their performance on a target domain, either without any additional target training examples (generalization) or with additional target examples (transfer). They found that current models over-fit to the particular training set and generalize poorly even to similar datasets. Besides, they also investigated the effect of knowledge transfer based on domain-level similarities and fine-tuning. Despite some improvements, the shared knowledge among multiple datasets of source domains has not been fully exploited. Although each MRC dataset involves the use of different expertise, there is still substantial overlap in the abilities required to answer questions across these datasets. Therefore, we believe it is meaningful to effectively exploit the shared knowledge among multiple datasets, and use them to promote the language understanding of the low-resource setup.

For a target domain, the learning objective is to find an optimal model that is well adapted to the current data. However, due to the lack of sufficient training samples, the optimal parameters oscillate with time during training. Therefore, it is difficult to learn an acceptable model with insufficient data. To address this issue, we propose an adaptive meta-learning framework for multi-source domain adaptation of MRC. The intuition behind the framework is that the past experience over multiple source domains can be used to facilitate the adaptation to a new environment. Specifically, we employ a model-agnostic meta-learning (MAML) algorithm [3] to aggregate the meta-knowledge from multiple source domains. The main focus of this paper is not to propose a new meta-learning algorithm, but to explore whether the representative meta-learning algorithm MAML is suitable for many-to-one machine reading comprehension scenario, and how to improve the domain transfer ability under the MAML framework.

Since meta-knowledge is transferred, there is still a gap between source domains and target domains. Especially when
the difference of data distribution is large, the performance of domain adaptation will be highly affected. To fill the domain gap, one possible way is to generate adversarial examples through adding imperceptible perturbations on inputs, which can be used to improve the model robustness [4]. However, the traditional adversarial training adds noises with the gradient of the source domain, which is futile to cross the domain gap. It may even broaden the gap. To solve this issue, we design an adaptive meta-learning framework based on adaptive adversarial training strategy, and employ gradient of the target domain to guide adversarial examples generation in the source domain. Therefore, we can obtain domain-specific meta-knowledge, rather than the fixed one.

The main contributions can be summarized as follows:

- To overcome the challenge of low-resource MRC, we propose an adaptive meta-learning framework. With this framework, the meta-knowledge across multiple source domains can be learned, which is further used to enhance the target-domain MRC ability. Besides, it can be easily combined with arbitrary MRC models like BERT.
- To bridge the gap between domains in meta-learning, we devise an adaptive adversarial training strategy. It helps obtain the domain-specific meta-knowledge. For both the generality and the individuality are taken into consideration, it is more suitable for each target domain.
- We evaluate our model on three resource-rich source datasets and five resource-poor target datasets, and the results demonstrate that the proposed model consistently outperforms state-of-the-art baselines considerably.

II. METHODOLOGY

A. Task Definition

In general, MRC tasks can be categorized into the extractive setting and the multiple-choice setting. To facilitate knowledge transfer, we formulate all the MRC datasets to the extractive type. Therefore, the example in all datasets contains a question, a paragraph, and an answer whose content is a local context extracted from the original paragraph.

Formally, given the paragraph \( P \) and a question \( Q \), the goal of MRC is to extract an answer span \([s, e]\) from \( P \), where \( s \) denotes the start position and \( e \) denotes the end position in \( P \).

B. Framework of Adaptive Meta-Learning

Our adaptive meta-learning framework consists of two parts: Multi-source Meta-Learning (MML) and Adaptive Adversarial Training (AAT). In the process of MML, the meta-knowledge is learned on the merged data of multiple source domains \( D^S = \{D^S_1, D^S_2, ..., D^S_N\} \) through MAML [3].

In the process of AAT, the adaptive adversarial examples are generated, which can be used in MML to encourage the meta-model to learn toward the target domain. Specifically, the generation of adaptive adversarial samples \( D^T_1 \) is guided by the gradient of the target domain \( \{D^T_1, D^T_2, ..., D^T_M\} \), respectively. Then, \( D^T_1 \) participates in the meta-learning process together with \( D^S \) in a data-augmentation way. The details of adaptive meta-learning are summarized in Algorithm 1.

Algorithm 1 The procedure of Adaptive Meta-Learning

Input: Multi-source data \( D^S \), target data \( D^T \), three sets of parameters \( \theta^*, \theta, \theta_t \), and their learning rate \( \alpha, \beta, \gamma \)
Output: Optimized parameters \( \theta^* \)
1: Initialize \( \theta^*, \theta_t, \theta \);
2: Fine-tune the target model parameters \( \theta_t \) on data \( D^T \):
   \( \theta'_t = \theta_t - \gamma \nabla_{\theta_t} L(f(\theta_t, D^T)) \);
3: for all \( X^S \) from \( D^S \) do
4:   Compute gradients on \( \theta_t \) with data \( X^S \), and generate adversarial samples:
   \( \tilde{X}^S = X^S + \lambda \nabla_{\theta_t} \theta \);
5: end for
6: Merge \( D^S \) and \( D^T \), get new source data \( D^S_t \);
7: while not convergent do
8:   Sample data \( \{D^S_1, ..., D^S_N\} \) from \( D^S_t \);
9:   for \( i = 1 \) to \( K \) do
10:   Sample support set \( D^S_{Sup} \) from \( D^S_i \) and train the model \( \hat{\theta} \):
11:   \( \hat{\theta} = \hat{\theta} - \alpha \nabla \hat{\theta} L(f(\hat{\theta}, D^S_{Sup})) \);
12: end for
13: Meta model update:
   \( \theta^* = \theta_{acc}/K, \theta^{*t} = \theta^* - \beta \nabla \theta_{acc}/K \);
14: end while
15: Transfer \( \theta^* \) as the initial parameters to the downstream task \( D^T \) and fine-tune.

C. Multi-source Meta-Learning

MML is used to aggregate the meta-knowledge from multiple resources. In the meta-learning problem setting, the goal is to learn models that can learn new tasks from small amounts of data. To achieve this, meta-learning algorithms require a set of meta-training and meta-testing tasks drawn from a certain distribution. The key assumption of learning-to-learn is that the tasks in this distribution share common structure that can be exploited for faster learning of new tasks.

There are two sets of model parameters required in meta-learning. One set of parameters is used for the meta-training task denoted as \( \hat{\theta} \), another set is used for the meta-testing task denoted as \( \theta^* \). For a downstream task, \( \theta^* \) is employed for model initialization and to be fine tuned afterward. The process of MML is described as follows:

Meta-learning involves multiple sampling data subsets \( D^S_{Sample} = \{D^S_1, D^S_2, ..., D^S_N\} \). Each sampling subset \( D^S_i \) includes a support set and a query set, denoted as \( D^S_i = \{D^S_{Sup}, D^S_{Que}\} \), and each pair of the support set and the query set is sampled from the same distribution.

Fast Optimization In a sampling subset, we utilize the support set \( D^S_{Sup} \) to optimize \( \hat{\theta} \). This process can be regarded as applying \( D^S_{Sup} \) as the training data to train a model, and the model parameters are optimized by stochastic gradient descent.
Fig. 2: Overview of the adaptive meta-learning framework. The larger circles with solid line represent source domains, and the smaller brimless circles represent target domains. Adaptive meta-learning is conducted by adding adaptive adversarial examples into multi-source meta-learning.

\[ \hat{\theta}' = \hat{\theta} - \alpha \nabla_{\hat{\theta}} L D_{Que}^S(f(\hat{\theta})), \]

where \( L \) denotes the cross-entropy loss function, and \( \alpha \) denotes the learning rate.

**Gradients Calculation** After the previous step, we can obtain a set of optimized parameters. Then, we utilize these parameters to calculate the gradients on the respective query set. This process can be regarded as applying \( D_{Que}^S \) as the development data to evaluate the sampled task.

**Gradients Accumulation** Next, the gradients of multiple sampled data points are accumulated. Since the accumulation operation considers the gradient direction generated from various subsets, it can be regarded as the shared knowledge across multi-domain being learned.

**Parameters Update** The cumulative gradient is used to update the parameters of the meta-model. Each update of \( \theta^* \) is seen as one step of training for meta-model. The formula used to update the parameters is as follows:

\[ \theta^* = \theta^* - \beta \nabla_{\theta_{acc}} / K, \]

where \( \theta_{acc} \) denotes the accumulative gradients on the parameters \( \theta \), and \( \beta \) is the learning rate, and \( K \) is the number of sampling tasks.

**Target Fine-tuning** Through multiple iterations of the above steps, a set of meta-model parameters \( \theta^* \) is learned. Then, we fine-tune the target model with the initial parameters \( \theta^* \) and the target dataset \( D_t^T \), to enable the model to be close to the target domain.

**D. Adaptive Adversarial Training**

Under the framework of meta-learning, we further propose Adaptive Adversarial Training (AAT) aiming to bridge the gap between domains. It is based on the idea of improving model generalization capability by adversarial training. Adversarial training uses a mixture of adversarial and original examples to train a model to make it more robust. Here, the adversarial examples are in the form of continuous vectors instead of actual texts in natural languages. Specifically, adversarial examples are slightly different from original examples, and they are generated by adding imperceptible perturbation to the embeddings of the original examples. We first feed the source data \( D_S^S \) into a frozen well-trained target domain network, and then generate adversarial samples \( D_{Tt}^S \) in the direction of the target domain. The process of AAT is described as follows:

**Perturbator Training** To generate adversarial samples more adapted to the target domain, we take the target model as a perturbator. After fine-tuning on the target data \( D_t^T \), we acquire a target model \( \theta_t \), which is used to generate directional perturbations. The perturbator is optimized as:

\[ \theta_t' = \theta_t - \gamma \nabla_{\theta_t} L(f(\theta_t, D_t^T)), \]

where \( \gamma \) is the learning rate and \( L \) denotes the cross-entropy loss function in MRC.

**Adversarial Examples Generation** Next, the perturbator \( \theta_t \) and the multi-source data \( D_S^S \) are used to generate the adaptive adversarial examples. Specifically, the objective of this step is to increase the loss of the target model, given \( X_i \) sampled from \( D_S^S \) and its added perturbation generated from \( D_t^T \). Through multiple iterations of the above step, we can obtain the set of adaptive adversarial examples \( D_{Tt}^S \) corresponding to \( D_S^S \). Then, \( D_{Tt}^S \) is used to train the meta-model in MML, as well as the previous source-domain data.

III. EXPERIMENTS

**A. Datasets**

In the experiments, we choose eight reading comprehension datasets with different sizes. According to the size of each dataset, we briefly divide the datasets into source domain and target domain. Specifically, three larger ones SQuAD [5], NewsQA [6] and HotpotQA [7] are merged into multi-source data, and five smaller ones DROP [8], WikiHop [7], ComQA [9], ComplexQuestion [10] and BoolQ [11] are regarded as target dataset respectively.

Our adaptive meta-learning method transfers the meta-knowledge aggregated from multi-source domain datasets to each target dataset. Due to the insufficient computation resources, we randomly sample 100K examples from each source dataset and combine them as the whole source-domain dataset.

**B. Baselines**

We mainly compare our method with several popular pre-trained models, which present state-of-the-art performance in recent MRC tasks. These pre-trained models are directly fine-tuned on target dataset. Besides, we also compare our method with the single-source transfer method and the multi-source transfer method. For all methods, we adopt the Exact Match (EM) and the F1 score as our evaluation metrics.

**BERT, XLM, XLNet, RoBERTA**: They are Transformer-based pre-trained language models.
SynNet: SynNet [12] is a data augmentation method using a two-stage synthesis network for MRC.

IMM: IMM [13] is an iterative multi-source mutual knowledge transfer framework for MRC.

MADE: MADE [14] is to model multi-dataset question answering with an ensemble of single-dataset experts, by training a collection of dataset-specific adapter modules that share an underlying Transformer model.

SingleSource-T: We use BERT to train a MRC model on single-source dataset, and then transfer the trained model to target domain and fine-tune it with target dataset.

MultiQA: MultiQA [2] is a BERT based MRC model, which is trained on merged multi-source dataset, and then transfer the trained model to the target domain.

SingleSource-M: We use MAML to learn meta knowledge on single-source dataset, and then transfer the meta knowledge to target domain and retrain it with target dataset.

MultiQA-M: We use MAML to learn meta knowledge on merged multi-source dataset, and then transfer the meta knowledge to target domain and retrain it with target dataset.

C. Experimental Setup

In the experiments, for each target domain and each baseline model, only 300 labeled samples are used for domain adaptation. For the MRC task, it can be seen as a low-resource scenario, because a few hundreds of samples are insufficient to learn a robust model. In the MADE [14], all the 300 examples are used to fine-tune the adapters. In the meta-learning, the 300 samples are divided into support set and query set. The number of the support set and the query set are set to 280 and 20 respectively. We implement our model based on the Transformers released by Hugging Face, running on a TITAN RTX GPU. The reported experimental results of our model are average of 5 runs. Following the settings of MultiQA [2], we set the training batch size to 24 and three learning rate $\alpha$, $\beta$, $\gamma$ uniformly to 3e-5. We set maximum length of input sequence to 384. The length of some documents exceeds the limitation, thus we tailor the original document into pieces with the sliding window size of 384 and sliding stride of 128. All models adopt Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$ for optimization and generally converge after 2 epochs.

D. Overall Performance

In order to evaluate the effectiveness of the proposed method, we test various baselines on five target datasets. Table I presents the main experimental results. From Table I we can make three important observations.

First, the models whose knowledge transferred from multiple source datasets perform better than those transferred from the single dataset in most cases. For the BERT-based (non-meta) knowledge transfer methods, MultiQA performs consistently better than NewsQA-T, HotpotQA-T and SQuAD-T, on EM and F1 respectively. For the meta knowledge transfer methods, MultiQA-M achieves a notable performance gain against NewsQA-M, HotpotQA-M and SQuAD-M. This is in accordance with our intuition, which indicates that there are shared abilities required to answer questions across different datasets, and exploring the shared abilities from multiple source datasets instead of the single one is helpful to improve the performance of the target-domain MRC.

Second, MultiQA-M performs better than the fine-tuned methods, including BERT, XLM [15], XLNet [16] and Roberta [17], and also outperforms the traditional knowledge transfer methods, including SynNet, IMM and MultiQA. The reason for the unsatisfactory results of fine-tuned methods mainly lies in the lack of sufficient training data for the target domain, and high-capacity parameters of the pre-training models. In general, pre-trained representations improve generalization, but their effect is moderate when the training data is insufficient. Besides, due to the meta-knowledge over multiple source domains promoting the fast convergence and adaptation to the target domain, transferring meta-knowledge is superior to the traditional knowledge. It suggests that meta-learning over multiple source domains is good at capturing the shared knowledge regardless of the data imbalance and different focus.

Third, our model achieves the best performance on all target datasets, and outperforms the baselines by a large margin. It indicates that not only the meta-knowledge but also the domain-specific meta-knowledge contribute to the effectiveness of our model. From the experimental results, we can infer that, in view of the meta-knowledge, there is still a gap between source domains and target domains. And correspondingly, the proposed adaptive meta-learning strategy is effective to fill the gap. Because different meta-knowledge can be obtained in the direction of different target domain under our design, the domain-specific meta-knowledge is well adapted to each target domain. In general, our design takes both generality and individuality into account, which is more appropriate for multi-source domain adaptation in MRC.

As shown in Table I our model and MADE model have different strengths on the target dataset domains. In fact, the two models follow two different technique lines, thus having

<table>
<thead>
<tr>
<th>Model</th>
<th>Drop</th>
<th>WikiHop</th>
<th>ComTransQA</th>
<th>ComplexQ</th>
<th>BoolQ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EM</td>
<td>F1</td>
<td>EM</td>
<td>F1</td>
<td>EM</td>
</tr>
<tr>
<td>BERT</td>
<td>46.51</td>
<td>54.04</td>
<td>19.29</td>
<td>22.27</td>
<td>12.25</td>
</tr>
<tr>
<td>XLM</td>
<td>48.16</td>
<td>57.94</td>
<td>16.20</td>
<td>21.22</td>
<td>12.96</td>
</tr>
<tr>
<td>XLnet</td>
<td>44.00</td>
<td>53.20</td>
<td>16.14</td>
<td>19.16</td>
<td>8.24</td>
</tr>
<tr>
<td>Roberta</td>
<td>48.05</td>
<td>56.51</td>
<td>17.04</td>
<td>20.11</td>
<td>10.01</td>
</tr>
<tr>
<td>MADE</td>
<td>47.91</td>
<td>55.93</td>
<td>21.49</td>
<td>25.91</td>
<td>21.08</td>
</tr>
<tr>
<td>NewsQA-M</td>
<td>48.15</td>
<td>56.09</td>
<td>23.31</td>
<td>27.64</td>
<td>28.15</td>
</tr>
<tr>
<td>HotpotQA-M</td>
<td>44.00</td>
<td>53.20</td>
<td>16.14</td>
<td>19.16</td>
<td>8.24</td>
</tr>
<tr>
<td>SQuAD-T</td>
<td>39.03</td>
<td>48.73</td>
<td>20.00</td>
<td>23.82</td>
<td>25.20</td>
</tr>
<tr>
<td>IMM</td>
<td>46.29</td>
<td>54.73</td>
<td>20.00</td>
<td>23.82</td>
<td>25.20</td>
</tr>
<tr>
<td>MADE</td>
<td>48.65</td>
<td>56.69</td>
<td>31.06</td>
<td>35.12</td>
<td>46.20</td>
</tr>
<tr>
<td>NewsQA-M</td>
<td>47.91</td>
<td>55.93</td>
<td>21.49</td>
<td>25.91</td>
<td>21.08</td>
</tr>
<tr>
<td>HotpotQA-M</td>
<td>47.52</td>
<td>55.40</td>
<td>21.78</td>
<td>25.79</td>
<td>24.26</td>
</tr>
<tr>
<td>SQuAD-M</td>
<td>48.65</td>
<td>56.09</td>
<td>23.31</td>
<td>27.64</td>
<td>28.15</td>
</tr>
<tr>
<td>MultiQA-M</td>
<td>52.99</td>
<td>60.34</td>
<td>25.45</td>
<td>29.98</td>
<td>33.45</td>
</tr>
</tbody>
</table>

TABLE I: Exact Match and F1 Score on Five Target Datasets with Different Methods. For Each Target Dataset and Each Evaluation Metric.
different expertise. Specifically, MADE is based on adapters, which provide a parameter-efficient alternative for the full fine-tuning in which we can only fine-tune lightweight neural network layers on top of pre-trained weights. However, our method is based on meta-learning, which helps optimize parameters to give better predictions in shorter time. Though the performance of two models is almost comparable, our method has two advantages. First, our method is model-agnostic and can be easily equipped with various machine learning models, including but not limited to pre-trained language models. Second, our method can train a more generalized model, which facilitates the learning of a novel task. For example, MADE performs better on WikiHop, ComQA and ComplexQuestion, but performs dramatically worse on Drop and BoolQ. Maybe it is because the former ones are all collected from Wikipedia, resulting in the similar question answering process, whereas Drop is crowd-sourced and adversarially created, and BoolQ provides quite different question type. By contrast, our method performs relatively stable among these datasets.

Besides, it is worth noting that the results in Table I are lower than those reported in the Leaderboards. The reasons lie in two aspects. First and foremost, the goal of our work is not to design a highly complicated model or pre-processing technique to significantly surpass the existing methods. Instead, we hope to explore a direction for MRC domain adaptation and other similar transfer learning problems. In fact, some of the top ranked models on the Leaderboards are based on super large-scale language models, e.g., T5 and GPT3. Due to the limitation of computational capacity, we finally adopt the widely used Bert-base as our backbone model, but the proposed cross-domain method can be easily combined with the existing models. Second, the datasets used for the manuscript and the Leaderboards are different. In our work, for each target domain, only 300 labeled samples are used for domain adaptation, while for the Leaderboards all labeled samples could be used.

To better measure the impact of randomness on the dataset, we have additionally compared the randomly sampling 100K data with intentionally selected 100K data from the multi-source dataset. Specifically, the intentionally selected 100K samples are ranked alphabetically. Figure 3 presents the comparison results. As seen from Figure 3, our model performs basically equal on two kinds of datasets. It proves that our proposed method is less susceptible to the randomness on the quality of the combined multi-source dataset, to some extent.

E. Ablation Study

The main highlight of our work is the application of adaptive adversarial training in meta-learning for multi-source domain adaptation of MRC. To evaluate the effect of adaptive adversarial training, we compare our framework with its following variants: (1) Meta-learning without adversarial training. (2) Meta-learning with traditional adversarial training, i.e., the perturbations are added in the direction of the source domain. (3) Meta-learning with adaptive adversarial training, i.e., the perturbations are added in the direction of the target domain.

Owing to the limitation of space, Figure 4 only illustrates the comparison results on four target datasets. From Figure 4, we can observe that compared with general meta-learning, the introduction of traditional adversarial training does not improve performance. In fact, the effectiveness of adversarial training mainly owns to its ability of local smoothness around data points, and pushing decision boundary away from data points. However, the generated examples surround the original examples, which has only limited ability to fill the domain gap. In order to guide the direction of generating adversarial examples from source domains towards target domain,
we further propose adaptive adversarial training. Moreover, traditional adversarial training under transfer scenario may further widen the gap between the source domain and target domain, thus resulting in a performance drop. Overall, it is surprising to find that meta-learning with adaptive adversarial training significantly outperforms the meta-learning with or without traditional adversarial training. The reason is that the proposed adaptive adversarial training can guide the generation of adversarial examples to the direction of the target domain, then the model trained with these adversarial examples can boost the adaptation ability.

IV. RELATED WORK

a) Transfer learning for MRC: MRC is a challenging Natural Language Processing research field with wide applications [18]. To address the low-resource MRC problem, previous studies usually use pre-trained language models to enhance the performance of MRC models, such as BERT. Meanwhile, many efforts have been made to use transfer learning methods to improve MRC performance in target domain via datasets in source domain, among which fine-tuning is the most common technique [19].

Recently, [20] presented an alternate meta-learning framework for complex question answering over knowledge bases, which jointly and alternately optimized a retriever to select questions, and a programmer to produce an answer. However, this framework is designed for the scenario of question distributional bias within the same dataset instead of learning shared knowledge among multiple datasets. Differently, we aim to learn and transfer the shared knowledge among multiple MRC datasets through adaptive meta-learning, MADE [14] and our work share the similar motivation, but lie in different technique lines. MADE is based on adapter, which can be used as a parameter-efficient alternative to fine-tuning. We provide a investigation of meta learning combined with adaptive adversarial training, which is a missing part in the literature.

b) Meta-Learning: Meta-learning exploits inherent structures in data to learn more effective learning rules for fast domain adaptation [21]. To make the model sensitive to the new task, one popular direction of meta-learning is to train a meta-learner to learn how to update the parameters of the target model. [3] proposed Model Agnostic Meta-Learning (MAML) that constrains the learning rule for the model and uses stochastic gradient descent to quickly adapt networks to new tasks. This pattern has been widely used to adapt deep networks to resource-poor environments. S2A [22] first trained a retriever and a programmer separately, and then established a meta-learning task for questions and employed MAML to fine-tune the programmer, based on the similar samples found by the retriever. Unlike the above works, we apply MAML to the multi-domain MRC task with an adaptive adversarial training in its framework, and it encourages the learned knowledge to fit each target domain well.

V. CONCLUSION

In this work, we propose an adaptive meta-learning framework for multi-source domain adaptation of MRC. With this framework, we can learn the meta-knowledge, which implies the shared abilities of reading comprehension, across multiple source datasets. Furthermore, an adaptive adversarial training strategy is devised to obtain domain-specific meta-knowledge, which fills the gap between source and target domains. Extensive experiments demonstrate that our framework establishes a state-of-the-art baseline for low-resource MRC.

REFERENCES

Enhanced BERT with Graph and Topic Information for Short Text Classification

Tong Zhang, Ailing Tang, Rong Yan*
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—Short text classification is an important natural language processing task due to the prevalence of short text on the internet and social media platforms. In this paper, we propose a novel graph-based short text classification method named GBBM (Graph-BERT-BTM Model) that leverages the powerful representation ability of graph data to capture the structural features of short text. In this work, we incorporate topic information to enrich and expand the feature space for the short text and compare our proposed method on five publicly available short text datasets with five existing models. Experimental results indicate the superiority of our proposed method.

Index Terms—short text classification, graph data, topic model

I. INTRODUCTION

With the rapid development of the Internet, a large number of short texts are generated every second on social media platforms [1], and short text classification has become a pressing concern in the field of natural language processing (NLP). It is crucial for various applications due to short texts with valuable information, such as sentiment analysis [2], recommendation systems [3], and topic identification [4]. However, the restricted number of words in short texts limits their semantic and contextual information, rendering the short text classification a more formidable challenge than the long text counterpart. Recently, deep learning models have been widely used to address short text classification tasks, such as convolutional neural networks (CNN) [5] and recurrent neural networks (RNN) [6]. These deep learning models can capture semantic and syntactic information in local consecutive word sequences well. However they may ignore global word cooccurrence in a corpus which carries non-consecutive and long-distance semantics [7].

With the development of deep learning technology, Graph Neural Networks (GNNs) [8] have been the subject of extensive research and implementation. TextGCN [9] constructed a co-occurrence graph of words and documents, which is then transformed into a node classification task using text graph convolutional networks for text classification. However, this model heavily relies on convolutional feature learning and aggregation operations for graph representation learning. As the number of layers increases, it loses its capacity to learn features and fails to attain parallel training. Chen et al. [10] approximated the integral of the embedding function through Monte Carlo methods and implemented batch training to significantly enhance the efficiency of model training. However, their models do not fully incorporate the global structural features of short texts.

In this paper, we present a novel classification method named GBBM (Graph-BERT-BTM Model) that blends graph neural networks and topic models for feature fusion. Specifically, the method involves three key steps. Firstly, we construct a document-topic-word heterogeneous graph on the corpus, which harnesses the representation capacity of graph data to capture the structural characteristics of short texts and supplements them with topic information to expand the feature space. Next, we partition the graph data into subgraphs to enable parallel processing. We input the feature representation obtained from the batch graph data into BERT [11] and utilize an attention mechanism to learn the semantic features of the short text in order to reduce sparsity. Finally, we merge the highly correlated topic information obtained from the topic model to improve the accuracy and effectiveness of short text classification.

II. METHODOLOGY

In this section, we present the construction of a heterogeneous graph and subgraph sampling method based on the graph model. To address the limited content of short texts and the lack of semantic and contextual information, we propose to build a document-topic-word heterogeneous graph structure on the corpus, where documents, topics, and words are jointly learned to expand the features of short texts.

A. Building a document-topic-word heterogeneous graph

In the heterogeneous graph of short texts, there are mainly three types of nodes: documents, topics, and words. The relationship edge between document-word is ascertained using calculating the term frequency-inverse document frequency (TF-IDF) value of word nodes contained within the document, while the edge between words is fashioned based on Pointwise Mutual Information (PMI). Furthermore, this paper introduces a topic model to represent the relationship strength among
documents, topics, and words. The formal definition of the
weight value $A_{ij}$ of the edge between $i_{th}$ node and $j_{th}$ node
is shown below:

$$
A_{ij} =
\begin{cases}
PMI(i, j) & n_i, n_j \text{ are word nodes.} \\
TF-IDF_{ij} & n_i \text{ is doc node, } n_j \text{ is word node.} \\
\text{doc - topic}_{ij} & n_i \text{ is doc node, } n_j \text{ is topic node.} \\
\text{topic - word}_{ij} & n_i \text{ is topic node, } n_j \text{ is word node.}
\end{cases}
$$

where $n_i$ and $n_j$ are the nodes in the graph $G=(V,E)$,
$\text{doc}$, $\text{topic}$, and $\text{word}$ are the different types of nodes,
and $\alpha$ is a constant, usually set as $0.15$.

B. Subgraph Sampling Based on Text Graph

In this paper, we denote the input data as a graph $G=(V,E)$,
where $V$ and $E$ represent the node and edge sets in the
graph, respectively. For each node $v_i$, we learn the features
of itself and neighboring nodes within a specific range to form
a subgraph set without edges. Inspired by Zhang et al. [12],
we first calculate the intimacy matrix $S(i, j)$ between node
pair $(v_i, v_j)$ using PageRank [13]. Then, we adopt the top-$k$
intimacy sorting method to select the k-nearest nodes to $v_i$
as its contextual information, constructing a subgraph $g_i$ with $k+1$
nodes. By repeating this process for all nodes, the complete
graph can be represented as $G=(g_1, g_2, \cdots, g_V)$. The intimacy
matrix $S$ based on PageRank is defined as Eq.(1):

$$
S = \alpha \cdot (\mathbf{I} - (1- \alpha) \cdot \mathbf{A})^{-1}
$$

where the factor $\alpha \in [0,1]$, and it is usually set as $0.15$.

In this work, we introduce the adjacency matrix $\mathbf{A}$
and its corresponding diagonal matrix $\mathbf{D}$ of graph $G$, denot-
ed as $\mathbf{D}(i,j)=\sum_j A(i,j)$. $\mathbf{A}=\mathbf{D}^{-1}$ represents the column-
normalized adjacency matrix. We employ the top-$k$ intimacy
sampling method to select the closest neighboring nodes
around node $v_i$, forming the subgraph $g_i$ centered on $v_i$.
Finally, the input graph data is projected into a feature vec-
tor representation through network embedding to obtain the
original feature representation of the node.

C. Model architecture

GBBM achieves joint training of BERT and BTM on the
text graph as shown in fig. 1. GBBM contains four parts:
(1) a document-topic-word heterogeneous graph, (2) subgraph
sampling, (3) feature fusion based on BERT and BTM, and
(4) classifier. GBBM learns text graph features using only the
attention mechanism in BERT, without any graph convolution
or aggregation operations. Firstly, the original input text se-
quence is transformed into graph-structured data, allowing for
the full exploration of complex semantic information within
short texts. Additionally, topic modeling is utilized to extract
topic information from short texts, which is then used to assist
in constructing a text graph consisting of document, topic,
and word nodes. PageRank is then employed to learn node
features surrounding specific targets in the graph data, which
are subsequently partitioned into small batches of subgraphs.

For node $v_i$, its original feature $x_i$ is represented in one-hot
form, and a fully connected layer is used to transform the
original feature of $v_i$ into an embedding representation with
consistent dimensions $d_h$. The initial input vectors of all nodes
in subgraph $g_i$ can be represented by Eq.(2):

$$
H^{(0)} = [h_i^{(0)}, h_{i,1}^{(0)}, \ldots, h_{i,k}^{(0)}]^T \in \mathbb{R}^{(k+1) \times d_h}
$$

with using multiple layers to iteratively update the representa-
tion of nodes, the output of layer $l$ can be expressed as Eq.(3):

$$
H^{(l)} = \text{Transformer}(H^{(l-1)})
$$

where

$$
\begin{align*}
Q &= H^{(l-1)}W_Q^{(l)} \\
K &= H^{(l-1)}W_K^{(l)} \\
V &= H^{(l-1)}W_V^{(l)}
\end{align*}
$$

Fig. 1. The architecture of GBBM

In the above equations, $W_Q^{(l)}$, $W_K^{(l)}$, $W_V^{(l)} \in \mathbb{R}^{d_h \times d_e}$
denote the involved varibales, and $X_i \in \mathbb{R}^{(k+1) \times d_e}$ is the raw features
of all nodes in the subgraph $g_i$. 
All initial input vectors of nodes in the short text subgraph are fed into BERT, where the attention mechanism automatically learns short text features and acquires hidden semantic representations. Additionally, the preprocessed short text sequence is used for training BTM. The document topic distribution (DT) and topic word distribution (TW) of the short text can be represented through Eq.(5) and Eq.(6), respectively.

$$DT = \varphi_{b|z} = \frac{n_{b|z} + \beta}{\sum_{b} n_{b|z} + M\beta}$$ (5)

$$TW = \phi_{z} = \frac{n_{z} + \alpha}{|B| + K\alpha}$$ (6)

where \(n_{b|z}\) represents the number of occurrences of co-occurrence word pair \(b\) in topic \(z\), \(K\) is the number of topics, \(|B|\) is the total number of bitersms, and \(M\) is the number of non-repeating words in the entire corpus [14]. The parameters \(\alpha\) and \(\beta\) are the prior parameters of the topic model.

Finally, the semantic representation generated by BERT and the potential topic information obtained from BTM are integrated to form the final representation vector. A Softmax classifier is then used for training, and the category probability distribution of the fused document-level topic information is calculated through Eq.(7). It allows us to obtain the corresponding category for the input short text.

$$P(L_i|H_i, DT_i) = \frac{\exp(H_i, DT_i)}{\sum_{k=1}^{K} \exp(H_k, DT_k)}$$ (7)

III. EXPERIMENTS

In this section, we evaluate the accuracy, precision, recall, and F1 score of each model on the same set of short text datasets to ensure a fair comparison.

A. Datasets

In this paper, we use five datasets to compare and verify our proposed method.

- **Pascal Flickr**: It is a collection of 4,834 captions primarily used for evaluating short text clustering or classification tasks.
- **GoogleNews**: It is a collection of 11,109 news article titles and snippets.
- **Ohsumed**: It is a dataset in the field of topic classification, and contains 7,400 medical abstracts with 23 different labels.
- **TREC**: It is one of the mainstream datasets in the question-answering task. This article uses the TREC-6 version dataset, which includes 5,452 training and 500 testing questions.
- **SST**: The Stanford Sentiment Treebank (SST) dataset is an extended version of the sentiment analysis dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Pascal Flickr</th>
<th>GoogleNews</th>
<th>TREC</th>
<th>SST</th>
<th>Ohsumed</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextCNN</td>
<td>48.14</td>
<td>82.36</td>
<td>90.02</td>
<td>81.13</td>
<td>58.38</td>
</tr>
<tr>
<td>TextRNN</td>
<td>46.34</td>
<td>80.47</td>
<td>87.33</td>
<td>80.79</td>
<td>50.27</td>
</tr>
<tr>
<td>FastText</td>
<td>45.79</td>
<td>81.46</td>
<td>88.90</td>
<td>82.27</td>
<td>52.34</td>
</tr>
<tr>
<td>TextGCN</td>
<td>52.17</td>
<td>84.48</td>
<td>90.06</td>
<td>81.66</td>
<td>64.56</td>
</tr>
<tr>
<td>BERT-raw</td>
<td>55.10</td>
<td>87.82</td>
<td>95.33</td>
<td>89.67</td>
<td>68.55</td>
</tr>
<tr>
<td>GBBM+DT</td>
<td>59.09</td>
<td>92.64</td>
<td>97.00</td>
<td>93.67</td>
<td>71.61</td>
</tr>
<tr>
<td>GBBM+TW</td>
<td>58.95</td>
<td>90.67</td>
<td>97.33</td>
<td>91.35</td>
<td>70.22</td>
</tr>
</tbody>
</table>

MR. In this paper, we select SST-2 that contains 9,613 documents.

In accordance with standard practice, the datasets was split into training set, verification set and test set according to the ratio of 7:1.5:1.5.

B. Experimental Settings

In this work, the configuration of topic model employs the parameter settings from the original paper, specifically \(\alpha=50/k, \beta=0.01\). GBBM features a hidden layer size of 32, 2 hidden layers, 2 multi-head attention mechanisms, hidden layer dropout of 0.5, attention dropout of 0.3, and utilizes GULE function as its activation function. We assume that the node residual term is independent and determined solely by the original input feature \(R(H(k−1), X, G)=X\). An early-stop strategy is used during model training to avoid overfitting. We use NLTK⁶ to preprocess the short text data.

C. Results and Analysis

We compare our proposed method with five different models that have been used in recent years across five different classification tasks as shown in Table I. It is evident from the table that irrespective of the type of topic information fused, the proposed GBBM outperforms the other comparative models in terms of classification on all five datasets. Experiments show that the utilization of topic information from short texts not only serves as additional features to alleviate the issue of data sparsity during graph construction but also aids the model in quickly capturing crucial information from texts during the training phase.

<table>
<thead>
<tr>
<th>Model</th>
<th>P(%)</th>
<th>R(%)</th>
<th>F1(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>HUM</td>
<td>84.54</td>
<td>91.15</td>
<td>87.72</td>
</tr>
<tr>
<td>NUM</td>
<td>97.90</td>
<td>93.96</td>
<td>95.89</td>
</tr>
<tr>
<td>ABBR</td>
<td>100.00</td>
<td>97.12</td>
<td>98.54</td>
</tr>
<tr>
<td>ENTY</td>
<td>85.38</td>
<td>84.73</td>
<td>85.06</td>
</tr>
<tr>
<td>DESC</td>
<td>100.00</td>
<td>83.23</td>
<td>90.11</td>
</tr>
</tbody>
</table>


To further demonstrate the efficient text classification performance of GBBM, we present a comparison of classification
measures for different categories of the TREC dataset in Table II. The experiment shows that the short text graph established based on document-topic-word nodes enables GBBM to have more available contextual information. Even with only 16 texts in the DESC category of the TREC dataset, GBBM can still efficiently complete short text classification tasks.

Table III presents the experimental results of GBBM incorporating multiple perspectives of topic representations, where GBBM+DW indicates GBBM that integrates document-word topic information and GBBM+DT+TW represents GBBM that jointly learns document-topic-word features. As shown in Table III, it can be observed that GBBM+DT+TW and GBBM+DW perform better on most datasets, indicating that the joint learning of document-topic-word features can effectively capture the topic information, thereby improving the classification performance.

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>CLASSIFICATION PERFORMANCE OF GBBM WITH MULTI-ANGLE TOPIC REPRESENTATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PascalFlickr</td>
</tr>
<tr>
<td>BERT</td>
<td>55.10</td>
</tr>
<tr>
<td>GBBM+DW</td>
<td>57.13</td>
</tr>
<tr>
<td>GBBM+DT+TW</td>
<td>57.37</td>
</tr>
</tbody>
</table>

Furthermore, we analyze the learning performance of GBBM with different subgraph sizes (parameter s) on the TREC dataset as shown in fig. 2. The learning performance of GBBM reaches its best when s increases from 1 to 6. However, with a further increase in s, the performance sharply declines. Similar results are observed for other datasets, albeit with different optimal values of s.

![Fig. 2. The performance with different subgraph sizes](image)

(a) accuracy curve  (b) loss curve

IV. CONCLUSION

In this paper, we propose a novel approach to short text classification that integrates graph neural networks and topic models for feature fusion. We conducted experiments on five public real-world datasets and compared the performance of our method with the current state-of-the-art text classification models. The results showed that our model outperformed the other five baselines, demonstrating the effectiveness of our approach. In future research, we plan to enhance the filtering of weakly related node information based on topic features, while retaining deep semantic information, in order to mitigate the issue of limited memory due to a large number of nodes and improve the accuracy of short text classification.

ACKNOWLEDGMENT

This research is supported by the National Natural Science Foundation of China (Grant No. 61866029).

REFERENCES

Abstract—Few-shot link prediction is an important recent research direction in the field of knowledge graph. To learn better entity and relation representations, previous works utilize the neighborhood information of entities but they ignore the associated relations. More than this, the link involves few-shot entities cannot be well linked by these methods. Therefore, this paper proposes a novel method that dynamically aggregates local neighbourhood information from entities and relations and introduces the information bottleneck principle to filter irrelevant features to better capture the feature of few-shot entities. Experimental results on three benchmark datasets demonstrate that our model outperforms other models and that our approach can effectively improve the model performance on those entities in the tail of the long-tailed distribution.

Index Terms—Link Prediction, Variational Information Bottleneck, Heterogeneous Graph

I. INTRODUCTION

Knowledge graphs use a structural and easily exploitable way to store the knowledge which describes the real world. This allows real-world knowledge to be well applied to many downstream tasks such as intelligent question and answer (QA) systems, financial anti-fraud domains, and search engines. These structural facts are recorded and described by KG in the form of \((e_i, r_k, e_j)\), for example, (Bill Gates, livein, Washington), describing the fact that Bill Gates lives in Washington. Although KG contains many entities, relations and triples, there are still incompleteness in KG, which has motivated researches on link prediction.

In literature, many sophisticated models have been designed to complement the facts in KG [1]. Among them, representation learning [2] is one of the mainstream approaches. It aims to learn better representation of nodes to accurately predict missing entities. Due to the graph structure of KG, it is natural to introduce graph convolutional networks (GCN) into link prediction. In addition, the emergence of graph attention networks (GAT) [3]–[5], which incorporate attention mechanisms into GCNs, achieve great improvement.

However, as shown in Figure 1, when we utilize the neighborhood information to learn the representation of entities, we observe the following phenomena: (i) The relation “cooperate” links to different head entities, and it may be more important when linking to “Bill Gates” instead of “David”, because the former is a influential figure, which illustrates that the contribution of the same relation may be different due to different head entities. (ii) The tail entity “ChatGPT“ may be more important when linking to “develop” instead of “user_of”, because “develop” needs more effort, which illustrates that the contribution of tail entity may be affected by the relation. (iii) The entities and relations presents a long-tailed distribution, which makes the model is easily disturbed by irrelevant features because of lack of sufficient data. Based on these observations, we summarize two drawbacks of the previous representation learning approaches:

- When learning entity representations, various graph attention mechanisms are designed to measure the importance of neighboring entities, but ignore the fact that adjacency relations and neighboring entities can interact with each other and jointly determine the semantics of the current centric node.
- When extracting features of entities and relations, previous models take redundant features into account, though
they were not relevant to the link prediction task, which limited further improvements in model performance.

Therefore, to overcome these challenges, we propose a novel approach to simultaneously exploit neighboring entities and relations to obtain in-learning graph representations and add noise during the learning process to refine features useful for graph embedding while obtaining better robustness. Specifically, when learning graph embeddings, we design a novel type-aware graph attention mechanism to take neighbor entities and neighbor relations into account. Between the encoder and decoder, we introduce variational information bottleneck (VIB) [6], [7] to distill the latent representations output by the encoder, and add Gaussian noise in this process using the information bottleneck principle. Our contributions are as follows:

- We dynamically adjust the weights of relations and tail entities when aggregating information about neighboring nodes and let them optimize each other when updating node embeddings.
- We introduce the principle of information bottlenecks to control the information from the encoder to the decoder, which helps the model to better capture potentially useful features and improve its generalizability to few-shot entities.
- Experimental results on the FB15K-237, WN18RR and FB15K benchmark datasets show the strength of our model against another model in various metrics.

II. RELATED WORK

A. Link Prediction

Massive efforts have been put into research on link prediction. Among all the link prediction methods, the most successful ones learn expressive representation for entities and relations and predict missing entities in incomplete KG. These methods can be divided into three groups: translation-based models, neural network based model and GNN-based models. In this section, the three groups’ models will be reviewed.

The translation-based models represented by TransE [1] and TransH [2] treat $r_k$ in each triple $(e_i, r_k, e_j)$ as a translation operation from $e_i$ to $e_j$. Unfortunately, those models are not powerful enough to represent the complex facts in KG, so more and more sophisticated models have been designed for this challenge, such as TransD [8], TranSparse [9] and TransG [10]. In addition, RotatE [11] rotates entities through relations to achieve translation operations. These methods perform a series of subtle operations on entities and relations in combination with the actual situation, and effectively improve the accuracy of link prediction.

Due to the excellent performance of neural network models, neural networks are gradually being used in this task. ConvE [12] scores all candidate entities through convolution layer and dot layers, and ConvKB [13] considers global information on this basis. It is worth mentioning that ParamE [14] considers relations as parameters of neural networks to train models and complete link prediction tasks. However, none of these models were aware of the importance of graph structure information. Therefore, GNN-based model gradually attracts the attention of more researchers. R-GCN [15] first applies the graph convolution operation to modeling relational data for link prediction. Latterly, KGBAT [16] introduced the attention mechanism to integrate neighborhood information to improve embedding quality. After that, researchers conducted a lot of research on this basis. Zhang, Zhuang et al. [4] transforms it into relation-level attention and entity-level attention. Zhao, Zhou et al. [17] adds additional global attention mechanism. Fang, Wang et al. [18] consider the impact of neighborhoods on representation learning of relations. However, those GNN-based methods cannot solve the few-shot entities problem, and they do not give enough consideration to neighborhood relations when aggregate information.

B. Information Bottleneck

The starting point of information bottleneck (IB) principle [19] is to transmit as much information as possible to the output $y$ while compressing the input $x$ in some way. The IB framework has been widely concerned in deep learning to get brief but comprehensive information. Recent works use VIB to improve the interpretability of the model [7] use it to reduce the overfitting on low-resource target tasks. Their research shows that the IB framework has significant advantages in capturing general features and can effectively enhance the robustness of the model in the case of few-shot learning.

Inspired by these ingenious algorithms, we propose the model ViHAN(Variational Information Bottleneck for Heterogeneous Attention Network) which can consider the impact of neighborhoods on the importance of relations and entities and can also filter out the useless features that the general-purpose feature extractors will inevitably extract.

III. PROBLEM STATEMENT

A large number of triples are stored in the KG, and every triple $\tau = (e_i, r_k, e_j)$ consists of a head entity $e_i$, a tail entity $e_j$ and a relation $r_k$. Following the work of Bordes [1], we define link prediction as the task of analyzing useful information and predicting missing entity given the relation and its head entity. For example, for input $\tau = (\text{France, capitalis, ?})$, the role of the link prediction model is to analyze KG and output the entity that the symbol "?" is most likely to refer to. During prediction, the model ranks the entities in the candidate set and selects the most likely correct entity [20] as the answer to what is the capital of France.

IV. OUR MODEL

In this section, we will detail our ViHAN Model based on encoder-decoder architecture. Figure 2 shows the overall architecture of the model. Specifically, for each triple in KG, we dynamically calculate the importance of the relation and tail entity based on the inherent characteristics of the triple. According to the results, we aggregate the neighborhood information of the central entity to enhance the representation of entities. In order to eliminate useless features to improve robustness of model, we introduce VIB to filter features and
help model learn better representation of few-shot entities. Afterwards, ConvKB analyze the global embedding properties of the triple with the help of score function to predict missing facts.

A. Joint Entity and Relation Encoder

In this section, we give detailed descriptions of our heterogeneous attention networks and the variational information bottleneck.

1) Heterogeneous Attention Networks: Intuitively, not all neighbors are equally important for representing the central entity, and the weights of neighbor entities and relations should be queried in combination with the graph structure information. Under the above consideration, we propose a new neighborhood-aware attention mechanism for link prediction. We add an additional query vector \( q \) to each node, which is used to query their importance in the current triple \((e_i, r_k, e_j)\). The query method is as follows:

\[
W_{(k,i)} = q_i^T q_k, W_{(j,k)} = q_j^T q_k
\]

where \( W_{(k,i)} \) refers to the importance of relation \( r_k \) queried from head entity \( e_i \) in current triple. Similarly, \( W_{(j,k)} \) refers to the importance of tail entity \( e_j \) queried from relation \( r_k \). \( q \in \mathbb{R}^{1 \times D} \) is the additional query vector used to calculate the relative attention score between two nodes and \( D \) is the dimension of our input embedding. After querying the importance of the relation and tail entity, to enrich the semantic representation, we integrate the neighborhood information in the current triple as Equation 2:

\[
\alpha_i = W^H (e_i||\sigma (W_{(k,i)} r_k)||\sigma (W_{(j,k)} e_j))
\]

where \( \alpha_i \) is the semantically integrated representation of current triple, \( W^H \) is training parameter used to collect neighborhood information, \( \sigma \) is the activation function like tanh, and \( "||" \) refers to the concatenating operation.

Then, we need to aggregate the information from different triples into the representation of each entity. Because different triples should be given different degrees of attention, we need to calculate the relative attention score \( \eta \) of each triple first. The process is shown in Equation 3:

\[
\eta_i = \text{softmax}(\sigma(W^R \alpha_i)) = \frac{\exp(GELU(W^R \alpha_i))}{\exp(\sum_{\chi_i \in \chi} \sigma(W^R \alpha_i))}
\]

where \( \chi_i \) is the triples with the head entity \( e_i \), \( \chi \) denotes the triple set in KG. After using softmax, activation function and weight matrix \( W^R \), the neighborhood information from different triples should weight fused. Hence, we use the relative attention scores \( \eta \) and the triple information \( \alpha \) to obtain the entity embedding \( \hat{e} \):

\[
\hat{e}_i = BN(\sum_{\chi_i \in \chi} \eta_i \alpha_i)
\]

We calculate the importance weight of local neighborhood information and aggregate that information into the representation of entities. Then, we concatenate the embedding from different heads to extract diverse features as Equation 5:

\[
\hat{e}_i = \|M_{m=1} BN(\sum_{\chi \in \chi} m^m \eta_i^m \alpha_i^m)
\]

Where \( M \) is the number of multiple heads and we use two heads in our code. Now \( \hat{e}_i \in \mathbb{R}^{2d} \) and \( r \in \mathbb{R}^d \) are not consistent in dimension. In order to ensure dimensional consistency, we also perform a linear transformation on the embeddings of relations via a matrix \( W^r \in \mathbb{R}^{D \times 2D} \) as follows:

\[
\hat{r}_i = r W^r
\]

Now entities already contain a large amount of information from the neighborhood but lack valuable information from their own. So we add their initial entity embedding information via a weight matrix \( W_e \in \mathbb{R}^{D \times 2D} \) in the following way:

\[
\hat{e}_i = e W^e + \hat{e}_i
\]

2) Variational Information Bottleneck: Such general-purpose feature extractors will inevitably extract features irrelevant to the target task, which is particularly obvious on few-shot entities and relations. Therefore, we introduce this component to control the representation from encoder, aiming to eliminate irrelevant features and redundant information. We compress the representation of entities and relations and extract the latent variables through two linear layers and two activation functions. After that, we use neural networks to fit the mean and variance of the input and add noise in the process of reconstructing the input:

\[
E = \{ \mu(\hat{e}_i) + \rho \Sigma(\hat{e}_i) \} \| \hat{e}_i
\]

\[
R = \{ \mu(\hat{r}_i) + \rho \Sigma(\hat{r}_i) \} \| \hat{r}_i
\]

where \( \rho \) represents the noise with Gaussian distribution, \( \mu \) and \( \Sigma \) represent the mean and variance respectively, and their dimensions are one quarter of the input. \( E \) and \( R \) is the final representations learned by our encoder.

3) Training Objective: Inspired by the idea of TransE [1], the triple \((e_i, r_k, e_j)\) are expected to satisfy \( \hat{e}_i + \hat{r}_k \approx \hat{e}_j \), so for each triple, we measure its score as follows:

\[
s_{\tau_{ijk}} = \| E_i + R_k - E_j \|
\]

where \( \tau = (e_i, r_k, e_j) \) is any triple including positive samples and negative samples. Absolutely, we want to score \( s_{\tau_{ijk}} \) as low as possible for positive samples and as high as possible for negative samples, so we use hinge-loss during training:

\[
L_G = \max \{ s_{\tau_{ijk}} - s_{\tau'_{ijk}} + \gamma, 0 \}
\]

where \( \gamma \) is a margin hyperparameter and \( s_{\tau'_{ijk}} \) is the negative sample. The goal of training is to make \( s_{\tau_{ijk}} \) as close to 0 as possible, while \( s_{\tau'_{ijk}} \) as close to the threshold \( \gamma \) as possible.

At the same time, the VIB should also be limited during training. We hope it can remain the information related to
Graph ConvNet

and negative samples, ConvKB defines an implausibility score:

\[ L_{\text{D}} = \beta KL [p_0 (E|f), r (E)] + \beta KL [p_0 (R|f), r (R)] + L_G \]

where \( \beta \) is a hyperparameter, which is used to ensure that the loss of prediction and the loss of compression is approximately equal. \( p_0 (\cdot) \) represents the estimate of the posterior probability of output when input is known. And \( r (\cdot) \) is the estimate of the prior probability \( p (\cdot) \).

B. Decoder

We tried different models as the decoder, such as TransE, ConvE and ConvKB [13], and finally selected the ConvKB model as our decoder with the best performance. Its convolutional layer uses multiple filters on triples to generate diverse feature maps. Since the samples consist of positive samples and negative samples, ConvKB defines an implausibility score:

\[ F_D (\chi_i) = W \cdot \| M_{m=1}^{\text{ReLU}} (E_i, R_k, E_j) * \zeta_m \| \]

where \( \zeta_m \) is the hyperparameter of the \( m \)th filter. We concatenate the outputs of all filters, and then use the matrix \( W \in \mathbb{R}^{1 \times MD} \) to do the dot product (symbol \( \cdot \)) to calculate the scores of triples. Finally, the Adam optimizer is used in our decoder:

\[ L_D = \sum_{\tau \in (\chi \cup \chi')} \log(1 + \exp(l(\tau) \cdot F_D (\tau))) + \frac{\lambda}{2} ||W||^2 \]

Where \( \chi' \) is the negative triple set. When \( \tau \in \chi, l(\tau) = 1 \), and when \( \tau \in \chi', l(\tau) = -1 \). The purpose is to enable the model to effectively distinguish positive samples and negative samples.

V. EXPERIMENTS

To demonstrate the superiority of the ViHAN model, we conduct extensive experiments on the link prediction task and further explore the generalizability of the model for rare entities in the long tail.

A. Datasets

We evaluate our model on three widely used benchmark datasets, FB15k-237, FB15k and WN18RR. FB15k is a subset of the KG Freebase, containing 14,951 entities and 1345 relations. FB15k-237 is the dataset which removed the reverse relations in FB15k and it contains 14541 entities and 237 relations. WN18RR is a subset of WordNet with 18 relations and 40,943 entities.

B. Evaluation Protocol

Our model scores each entity in the candidate entity set, ranks them according to their scores, and finally calculates various metrics such as mean rank(MR), mean reciprocal rank(MRR), the proportion of valid triples in the top-N ranks(hit@N). Therefore, we replace the original entities in triples with candidate entities, then score and rank them through the model.

C. Training Protocol

The training process of our model can be divided into two stages: (1) We train the encoder to obtain more expressive embeddings, and the hyperparameter \( \beta \) used to balance the
### TABLE I

**EXPERIMENTAL RESULTS.** **BOLD** INDICATES THE BEST RESULTS IN EACH METRIC.

<table>
<thead>
<tr>
<th>Methods</th>
<th>FB15k-237</th>
<th>WNI8RR</th>
<th>FB15k</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR H@1  H@3 H@10</td>
<td>MRR H@1  H@3 H@10</td>
<td>MRR H@1  H@3 H@10</td>
</tr>
<tr>
<td>DistMult [21]</td>
<td>0.199 28.1 30.1 44.6</td>
<td>0.444 41.2 47.0 50.4</td>
<td>0.798 - - 89.3</td>
</tr>
<tr>
<td>ComplEx [22]</td>
<td>0.194 2.8 29.7 45.0</td>
<td>0.449 40.9 46.9 53</td>
<td>0.692 59.9 75.9 84.0</td>
</tr>
<tr>
<td>ConvE [12]</td>
<td>0.225 31.2 34.1 49.7</td>
<td>0.456 41.9 47.0 53.1</td>
<td>0.657 55.8 72.3 83.1</td>
</tr>
<tr>
<td>RotatE [11]</td>
<td>0.238 24.1 37.5 53.3</td>
<td>0.476 42.8 49.2 57.1</td>
<td>0.699 58.5 78.8 87.2</td>
</tr>
<tr>
<td>KBGAT [16]</td>
<td>0.518 46.0 54.0 62.6</td>
<td>0.429 36.1 47.7 57.2</td>
<td>0.868 82.4 90.2 93.5</td>
</tr>
<tr>
<td>ParamE-Gate [14]</td>
<td>0.399 31.0 43.8 57.3</td>
<td>0.489 46.2 50.6 53.8</td>
<td>0.813 76.6 85.0 89.6</td>
</tr>
<tr>
<td>DualE [23]</td>
<td>0.365 26.8 34.1 49.7</td>
<td>0.462 44.4 51.3 58.4</td>
<td>0.826 76.2 87.9 92.4</td>
</tr>
<tr>
<td><strong>HAN w 1 head</strong></td>
<td>0.495 41.7 53.2 63.7</td>
<td>0.439 36.5 48.4 57.2</td>
<td>0.886 85.9 90.6 92.7</td>
</tr>
<tr>
<td><strong>HAN w 2 head</strong></td>
<td>0.508 42.9 54.7 65.8</td>
<td>0.434 35.2 47.7 58.7</td>
<td>0.890 86.0 91.4 93.9</td>
</tr>
<tr>
<td><strong>HAN w 3 head</strong></td>
<td>0.514 45.2 54.1 63.5</td>
<td>0.449 36.5 50.0 60.0</td>
<td>0.889 85.9 91.1 93.7</td>
</tr>
<tr>
<td><strong>ViHAN w 1 head</strong></td>
<td>0.528 45.2 56.6 66.5</td>
<td>0.459 37.9 50.4 59.9</td>
<td>0.853 80.3 85.2 92.8</td>
</tr>
<tr>
<td><strong>ViHAN w 2 heads</strong></td>
<td>0.567 50.8 59.4 67.8</td>
<td>0.474 39.8 51.7 60.9</td>
<td>0.889 85.9 91.1 93.7</td>
</tr>
<tr>
<td><strong>ViHAN w 3 heads</strong></td>
<td>0.548 48.4 57.6 66.8</td>
<td>0.462 38.4 50.8 59.9</td>
<td>0.896 86.7 91.7 94.1</td>
</tr>
</tbody>
</table>

loss of two components is set to $1 \times 10^{-4}$; (2) On this basis, we train the decoder to perform the link prediction task. We use the Adam optimizer to update the parameters. The epoch number of the encoder and decoder is set to 3,600 and 200.

### D. Results and Analysis

The experimental results of our models with different numbers of attention heads are shown in the table I. For FB15k-237 and FB15k, it can be concluded that our model has achieved the best results compared with the baseline model. This shows that our strategy, which dynamically assigns weights to entities and relations when aggregating neighborhood information and eliminates redundant feature, is correct.

Also, the result on WN18RR shows that our model outperforms other benchmark models on most metrics. At the same time, we notice that other models like DualE are about 20 points lower than our model on some metrics like Hit@3 on FB15K-237 but slightly improved over our model on WN18RR. Based on the difference between the two datasets, we believe that the idea of using neighbourhood information in our model enables us to perform well on FB15K-237 with many triples and few entities. However, it performs not good enough on WN18RR with few triples and many entities because of the relative lack of neighborhood information.

In order to further prove the effectiveness of our method, we select some few-shot relations that have relatively few occurrences in the training set and visualized the entities connected by them on the test set. Combined with Figure 3, it is not difficult to find that our model can better partition entities under the few-shot relations. For example, clubs such as “Watford F.C.” and locations such as "Prague" are well distinguished by our model, while TransE is not. We think the reason why TransE can’t distinguish these entities is that clubs can sometimes express the concept of geographical location. This shows that our method can learn high-quality representation of the entities under the few-shot relations.

### E. Ablation Study

We conduct ablation experiments to explore whether the VIB can help the model capture deep latent features and improve its robustness. We counted the frequency of each entity in the training set and sorted them according to the number of occurrences. We name the 10% entity with the least number of occurrences as tail-10 entities, and use the same method to obtain tail-20 entities. Then we build four new test sets. The head entities and tail entities in those test sets are tail-10 entities and tail-20 entities respectively. Since irrelevant and redundant features are eliminated by the component, the model should have better performance than before. The experimental

![Figure 3](image-url)
results prove this. The performance of the model with this component on all indicators has been significantly improved. Notably, the VIB component can compress and reconstruct the original representation. From this point of view, the lower the occurrence frequency of entities, the more important the role of the VIB. It is not difficult to see from the experimental results that our VIB significantly improves the generalization of few-shot entities. This shows that our hypothesis is completely correct.

VI. CONCLUSION

In this paper, we propose link prediction method for few-shot entities based on the information bottleneck principle and evaluate the representation capability of the model on link prediction task. Specifically, we design heterogeneous attention networks to model the importance of neighborhood information and better enhance the representation of the central entities. In addition, we introduce VIB to mine the deep features of nodes and improves the performance on few-shot entities. Extensive experimental results show that our approach outperforms other baseline models on mainstream datasets.

VII. ACKNOWLEDGEMENT

The research reported in this paper was supported in part by the Shanghai Science and Technology Young Talents Sailing Program Grant 21YF1413900; the Outstanding Academic Leader Project of Shanghai under the grant 20XD1401700; and National Key R&D Plan 2021YFC3300602.

REFERENCES


HGAPT: Heterogeneous Graph Augmented Prompt Tuning for Low-Resource Fake News Detection

Xingchen Ding, Fei Li, Chong Teng*, Donghong Ji
Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education,
School of Cyber Science and Engineering, Wuhan University, Wuhan, China
Email: {xingos,lifei, csnlp, tengchong, dhji}@whu.edu.cn

Abstract—The dissemination of disinformation on social media platforms has a significant impact on personal reputation and public trust. There has been a recent surge of interest in fake news detection. However, detecting low-resource fake news, particularly those pertaining to recent events that have not yet been disseminated by users and are typically in short text, remains challenging due to the lack of training data and prior knowledge. In this paper, we introduce a novel framework named the Heterogeneous Graph Augmented Prompt-based Tuning framework (HGAPT) that can leverage the metadata of news such as publisher and topic to construct a heterogeneous graph in same batch, which improve the performance of low-resource fake news detection. We have conducted extensive experiments on two low-resource fake news datasets that were collected from real-world sources. The results demonstrate that our proposed framework outperforms state-of-the-art methods, with superior detection performance at the zero-shot setting.

Index Terms—Low-Resource, Fake News Detection, Social Media, Prompt Tuning, Heterogeneous Graph

I. INTRODUCTION

With the advent of the Internet and social media platforms such as Twitter and Facebook, people now have convenient access to vast amounts of information. However, the widespread use of social media has also led to the proliferation of fake news, which can distort facts and spread misinformation, ultimately leading to negative consequences. For example, the dissemination of fake political news can erode public trust in both governments and journalism. To protect individuals, governments, and the news ecosystem from the negative impacts of fake news [1], it is crucial to develop automated techniques for detecting fake news on social media. Therefore, the field of fake news detection has emerged as an area of great interest with the potential to provide significant benefits. Initial research in fake news detection focused on identifying effective features from various sources, such as textual content, user profiling data, and news diffusion patterns. Linguistic features such as writing styles and sensational headlines [2], and lexical and syntactic analysis [3], were explored to differentiate between fake news and real news. However, these feature-based methods were biased, time-consuming, labor-intensive, and susceptible to user manipulation. Recent studies [4], [5], [6] have attempted to tackle these challenges by utilizing various neural networks to acquire sophisticated representations for detecting fake news. [7], [8] also utilize the technique of fine-tuning Pre-trained Language Models (PLMs) for fake news detection. However, these methods exhibit suboptimal performance in low-resource settings without a sizable qualified training corpus.

For low-resource fake news, during the initial stage of a breaking event, only a limited amount of relevant short news with no comments and related knowledge is typically posted on social media. Therefore, it is appropriate to consider the metadata of the news, such as the publisher, topic, news source, and so on. People tend to believe news from a trusted and authoritative source, or news shared by publishers with a good reputation. In addition, certain topics, such as politics or controversial issues, are more likely to be targeted by fake news creators who seek to spread false information for their own agenda, which makes us more skeptical about news related to these topics. Drawing inspiration from the aforementioned observation, we propose a comprehensive solution to tackle the challenge of detection of fake news on social media platforms in low-resource environments. Our proposed solution is the Heterogeneous Graph Augmented Prompt-based Tuning framework (HGAPT). Specifically, we leverage news metadata to construct a batch-wise heterogeneous graph. In this graph, each labeled news item is represented by a node and edges between the nodes denote the similarity between the news items. HGAPT establishes edges in the heterogeneous graph based on whether two news items share common attributes such as speaker or topic, and further regulates the similarity of the representations learned by PLMs between each pair of news items to conform with the edges of the relation graph. The scalability of HGAPT is noteworthy since the heterogeneous graph is constructed based on a mini-batch of sampled data points per iteration.

To verify the effectiveness of our HGAPT framework, We developed two low-resource fake news datasets, respectively gathered from the Liar dataset [9] and [www.politifact.com]. Our experimental results demonstrate that our model outperforms other methods in the task of detecting fake news under low-resource settings and zero-shot settings.

II. RELATED WORK

Fake News Detection. Initial research in the detection of fake news has primarily focused on developing effective features to distinguish between fake and true news. Researchers have
explored linguistic patterns, such as special characters and keywords [10], writing styles and sensational headlines [2], as well as lexical and syntactic features [3], and temporal-linguistic features [4] to identify fake news. In addition to linguistic features, some studies have also proposed a range of user-based features [10, 11], including factors such as the number of fans, registration age, and gender, as potential indicators of fake news. Nonetheless, the process of designing effective functions for fake news detection can be time-consuming and requires significant domain expertise.

To tackle the aforementioned challenges, a variety of neural network models have been employed by researchers. For example, recurrent neural networks (RNNs) [4], convolutional neural networks (CNNs) [5], graph neural networks (GNNs) [12], multitask learning [13] have been utilized to learn representations from news content or diffusion graphs. Fine-tuning methods of pre-trained language models (PLMs) have thus far provided straightforward yet robust baselines in fake news detection [7, 14]. However, this approach is hampered by the gap between the pre-training and fine-tuning stages, leading to performance bottlenecks. In contrast to the aforementioned methods that either do not employ PLMs or use them insufficiently, we leverage an effective technique, i.e., prompt learning, to guide fake news detection utilizing a PLM.

**Prompting for PLMs.** Prompt-based learning has become increasingly popular for extracting knowledge from large language models. This trend is evident in recent research studies [15, 16] that have focused on prompt-tuning, which has gained significant attention over the past few years. With the advent of GPT-3, prompt-based learning has relied on handcrafted prompts to achieve impressive performance. More recently, AutoPrompt [17] and LM-BFF [18] have proposed automatic prompt construction through generating discrete tokens. In contrast to previous work on fake news detection [19, 20], our prompt-based framework primarily focuses on fake news detection by exploiting relations among a limited number of labeled news, which represents a novel exploration of this challenging task in a low resource setting.

**Graph Structure Learning.** Graph structure learning techniques aim to learn the graph structure and node embeddings of input samples simultaneously [21]. Typically, these methods consist of two iterative steps: (i) estimating the adjacency matrix that represents the graph structure using node embeddings, and (ii) employing graph neural networks (GNNs) to obtain new node embeddings based on the updated graph. Recently, graph structure learning has been utilized to estimate heterogeneous graphs among samples for effective propagation of supervised information [22, 23]. These methods estimate heterogeneous graphs that encode the similarity between sample embeddings. In contrast, our HGAPT model gauges similarity between samples using news prediction vectors without the need for additional parameters.

### III. Problem Statement and Background

In this paper, we aim to address the problem of fake news detection by leveraging a pre-trained language model (PLM) and a few labeled news examples. In prompt-based tuning, each input sample consisting of a pair \((x_i, y_i)\) is transformed into a pattern-verbalizer pair (PVP) [24], denoted by \((p(x_i), v(y_i))\). The pattern mapping function \(p(\cdot)\) takes \(x_i\) as input and produces a cloze question with masks. For instance, given a single sentence represented as ‘'x\_i = [CLS] News [SEP], ’' we can map it to a cloze question as follows: ‘'p(x\_i) = [CLS] News. It was [MASK], [SEP], ’' where the tokens [CLS] and [SEP] serve as special start and end markers, respectively.

In the context of prompt-based tuning, the verbalizer function \(v(\cdot)\) maps the label \(y_i\) to tokens that represent its semantic meaning. For instance, in this paper, labels such as "pants fire/false/mostly false/half true/mostly true/true” are mapped to tokens such as "fabricated/false/inaccurate/dubious/credible/authentic”. Given a PVP, the representation of the input \(x_i\) is obtained by taking the token embedding \(h_i^{[MASK]}\) corresponding to the [MASK] token. The class prediction \(\hat{y}_i\) is a probability distribution over all possible class labels, with the probability of the ground truth label \(y_i\) given \(x_i\) estimated as follows:

\[
q(y_i \mid x_i) = \frac{\exp(w_v^T h_i^{[MASK]} \cdot x_i)}{\sum_{y \in \mathcal{Y}} \exp(w_v^T h_i^{[MASK]} \cdot x_i)}
\]

Here, \(w_v\) is the logit vector of token \(v\) in the vocabulary, and \(\mathcal{Y}\) denotes the set of all possible class labels. Let \(y_i\) be a one-hot vector with all elements being 0 except for the one corresponding to the ground truth class label \(y_i \in \{1, \ldots, C\}\). The model is trained by minimizing the cross-entropy loss \(L_{CE}\) defined as:

\[
L_{CE} = \frac{1}{N} \sum_{i=1}^{N} - \log (\hat{y}_i)^T y_i,
\]

where \((\cdot)^T\) denotes the transpose operation, and \(N\) is the total number of samples in the training set.

### IV. Proposed Method

In this section, we introduce the proposed HGAPT, as illustrated in Figure 1. Our approach leverages supervised signals from training news samples by constructing and learning on batch-wise heterogeneous graphs. This method effectively enhances the detection of fake news.

#### A. Mask Representation Augment

A well-known challenge in prompting is the need for a fixed number of positions for the label, e.g., a single mask is needed for words present in the dictionary such as Yes/No; however, we need multiple positions to predict more complex ones with multiple tokens such as Half true. The label inventory commonly contains words tokenised into multiple tokens. [24] proposed a simple verbalisation technique where the original labels are replaced with words that can be represented with a single token from the vocabulary, e.g., Half true → Dubious. However, this will lead to the degradation of the model’s performance for fake news detection, because of the loss of label information during the mapping process.
Here, we propose a simple, yet effective, approach to overcome this problem. We take the original label inventory and tokenise all words, as shown in [1]. In the original labels box, we see six labels for fake news detection tasks and their tokens – {‘pants’, ‘fire’}, {‘false’}, {‘mostly’, ‘false’}, {‘half’, ‘true’}, {‘mostly’, ‘true’}, and {‘true’}. For each token of a label, we extract the vector representation from the PLM’s token embeddings $v_{\text{T tok}} = \text{TokEmb}(L_i)$. Afterwards, we obtain the final label representation ($L_E$) using an element-wise averaging for all $v_{\text{T tok}}$ (see Eq. 3).

$$L_E = \frac{1}{N} \sum_{\ell=0}^{N} \text{TokEmb}(L_i) ; \forall L_i \in \{ \text{Labels} \}$$  (3)

Finally, to obtain the augmented MASK representation $\hat{y}$ for each example, we take the dot product between the MASK representation for the masked token position, and each of the $L_E$ vectors.

**B. Heterogeneous Graph Construction**

Given a mini-batch $\mathcal{N} = \{x_i, y_i\}_{i=1}^{N}$ comprising $N$ randomly sampled sequence-label pairs, whose indices are stored in $\mathcal{I} = \{1, \ldots, N\}$. Our objective is to leverage additional supervised information by constructing a heterogeneous graph model.

A heterogeneous graph, denoted as $G = \{V, E, F, R, \mu, \phi\}$, where $V$ represents the set of nodes, $E$ the set of edges, $F$ the set of node types and $R$ the set of edge types, where $|F| + |R| > 2$. Each node $v_i \in V$ is associated with a node type mapping function $\mu : V \rightarrow F$, and each edge $e_i \in E$ is associated with an edge type mapping function $\phi : E \rightarrow R$ . Let $G$ denotes the heterogeneous graph among the $N$ training samples in $\mathcal{N}$. In particular, $V$ is a set of nodes where each node $v_i \in V$ corresponds to one training sample $x_i$, $E = \{e_{ij}\}$ is a set of edges between the $N$ training samples, and $R$ is a set of relation types of news. Hence, we establish

$$e_{ij} = \begin{cases} x & \text{if } v_i \text{ is related to } v_j \\ 0 & \text{otherwise} \end{cases}$$  (4)

Where $x$ is selected from the set $\{0, 0.2, 0.5, 1\}$ and its value depends on the type of edge. In this paper, our primary focus is on utilizing news metadata, including class, topic, publisher, etc., to construct a heterogeneous graph as depicted in Figure 2.

**C. Heterogeneous Graph Learning**

On the heterogeneous graph $G$ of mini-batch $\mathcal{N}$, we decompose the original classification task into three sub-problems: (i) a node classification problem that aims to accurately classify each node into its corresponding class, (ii) an edge prediction problem that aims to establish connections between nodes belonging to the same class and disconnect nodes belonging to different classes, and (iii) a heterogeneous graph learning problem that aims to minimize the dissimilarity between the predicted graph and the original graph.

The problem of node classification is equivalent to the initial fake news classification task. As such, we can generate a class prediction $\hat{y}_i$ for $v_i$, which corresponds to $x_i$, by utilizing [1]. We can then calculate the loss $L_{\text{CE}}$ using [2].

In the edge prediction problem, we aim to predict the relationship between two nodes $v_i$ and $v_j$ by establishing $\hat{e}_{ij}$ based on the relevance between their corresponding representations $\hat{y}_i$ and $\hat{y}_j$:

$$\hat{e}_{ij} = g(\hat{y}_i, \hat{y}_j)$$  (5)

where $\hat{y}_i$ and $\hat{y}_j$ are obtained using the MASK Representation Augment module. In this paper, we use cosine similarity to compute the function $g(\cdot, \cdot)$, although other choices are possible. One could leverage auxiliary heterogeneous graphs or calculate $\hat{e}_{ij}$ based on representation similarity, such as $g\left(\textbf{h}_i^{\text{CLS}}, \textbf{h}_j^{\text{CLS}}\right)$ and $g\left(\textbf{h}_i^{\text{MASK}}, \textbf{h}_j^{\text{MASK}}\right)$. However, we prefer to use $\hat{y}_i, \hat{y}_j$ as they carry more semantic information that is relevant to each news, yielding better empirical performance.

To measure the loss of edge prediction, we use the $L_{\text{edge}}$ loss defined as follows:

$$- \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{A}(i)} e_{ij} \log (\hat{e}_{ij}) + (1 - e_{ij}) \log (1 - \hat{e}_{ij})$$  (6)

where $\mathcal{A}(i) = j \in \mathcal{I}$ and $i \neq j$ is the set of all nodes except $v_i$, and $e_{ij}$ is the true relationship between $v_i$ and $v_j$. Note that we
Algorithm 1 Heterogeneous Graph Augmented Prompt-based Tuning

**Input:** A small set of news \( C \).

**Output:** Assign news labels \( y \) to given unlabeled target data.

1. for each mini-batch \( N_i \) of the news \( C \) do:
   2. Pass \( N_i \) to the PVP and then PLM to obtain its [MASK] token representation \( \text{mask}_i \).
   3. Pass \( \text{mask}_i \) to the representation augmented module to obtain its class-level feature \( y_i \).
   4. Compute the classification loss \( L_{CE} \).
   5. for each relation \( \text{edge}_i \) of heterogeneous graph \( G \) do:
      6. for each event-level feature \( \hat{y}_i \) of mini-batch \( N_i \) do:
         7. for each event-level feature \( \hat{y}_i \) of mini-batch \( N_i \) do:
            8. Compute the loss of edge prediction \( L_{edge_i} \) as Eq[6]
      9. Compute the joint loss \( L_{edge} \) as Eq[6]
   10. Compute the joint loss \( L_{HG} \).
   11. Jointly optimize all parameters of the model using the loss \( L = L_{CE} + \alpha L_{HG} \).

Avoid introducing additional parameters to the function \( g(\cdot, \cdot) \) to reduce the risk of overfitting, given the limited number of labeled samples.

In order to address the problem of heterogeneous graph learning, we optimize the model to minimize the heterogeneous graph prediction loss \( L_{HG} \) for each mini-batch \( N_i \) as a whole. Here, \( f_i \in \{0, 0.5, 1\} \) represents a mapping of edge types to control the contribution of the corresponding \( L_{edge} \), and \( L_{edge} \) denotes the loss function for each edge type \( i \) in the heterogeneous graph. The expression for \( L_{HG} \) is given by:

\[
L_{HG} = \sum_{i \in F} f_i \cdot L_{edge_i}
\]

(7)

**D. Model Training**

We jointly train the model with the cross-entropy and supervised objectives:

\[
L = L_{CE} + \alpha L_{HG}
\]

where \( \alpha \) is a hyperparameter to control the contribution of this \( L_{HG} \). Algorithm 1 presents the training process of our approach.

V. EXPERIMENTS

In this section, we introduce the experiments to evaluate the effectiveness of HGAPT. Specifically, we aim to answer the following evaluation questions: **EQ1:** Can HGAPT improve low-resource fake news classification performance by exploiting the limited supervised information of news? **EQ2:** How effective are heterogeneous graph construction in improving the detection performance of HGAPT? **EQ3:** Can HGAPT improve the performance of zero-shot fake news detection task?

**A. Datasets and Experiment Settings**

Currently, there are no publicly available benchmarks for detecting fake news on social media in low-resource settings. In this paper, we utilize the original testing set from the LIAR dataset [9] for testing. Additionally, we randomly select 512 examples per class from the original training set and used them as the training set. Additionally, we create a new fake news dataset, called the PolitiFactFewShot dataset, using authoritative sources from www.politifact.com. Table I and Figure 3 provide detailed statistics for both datasets. Our evaluation metrics, accuracy and F1 score. We also transformed the original six-class datasets into two classes to verify the performance of our model in a general setting for detecting fake news, for instance, the label ‘pants fire/false/mostly false’ is categorized as ‘fake’, while ‘half true/mostly true/true’ is classified as ‘true’. We selected the hyperparameter \( \alpha \) from \([0 : 0.2 : 0.5 : 1]\), and the batch size from \([4 : 8 : 16]\) for our HGAPT model.

**B. Baseline Model**

We compare our proposed model with several state-of-the-art baseline methods, which are described as follows: 1) RNN [4]: This model is based on a recurrent neural network (RNN) with gated recurrent units (GRU) for learning relevant post features over time in rumor detection; 2) AttLSTM [26]: This is a long short-term memory (LSTM) model that uses attention mechanism to consider the importance of words in the relevant posts; 3) FNDML [13]: This model employs Multitask Learning (ML) methodologies to train reliable classifiers to detect fake news; 4) FT+ERINE [27]: We use an existing fine-tuning technique based on the ERINE pretrained language model; 5) FakeBERT [7]: This model combines different parallel blocks of the single-layer deep CNN having different kernel sizes and filters with the BERT; 6) ParallelBERT [14]: This model uses two parallel BERT networks to perform fake news detection. One of the BERT networks encodes news, and another encodes news-related knowledge; 7) PT-* [28]: We improve an existing prompt-based tuning technique on the ERINE PLM for fake news detection and extend it for our task. The \( * \) in PT- represents different extensions, including knowledgeable prompt learning (KPL) [20], supervised contrastive learning (SCL) [29], and our proposed Heterogeneous Graph Augmented (HGA) framework.

We evaluate these models in the most challenging setting of detecting fake news in low-resource settings, where news on social media is typically shorter in length and less frequent.

**C. Results**

In order to answer EQ1, we conducted a comparison of HGAPT with the baselines outlined in Section V-B for low-resource fake news classification. The performance of our
Table II. Test performance (%) measured on low-resource fake news datasets. The best results are highlighted in bold.

<table>
<thead>
<tr>
<th>Model</th>
<th>LiarFewShot</th>
<th>PolitifactFewShot</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6 classes</td>
<td>2 classes</td>
</tr>
<tr>
<td></td>
<td>Acc.</td>
<td>Mac-F1</td>
</tr>
<tr>
<td>RNN</td>
<td>0.254</td>
<td>0.237</td>
</tr>
<tr>
<td>AttLSTM</td>
<td>0.257</td>
<td>0.222</td>
</tr>
<tr>
<td>FNDFML</td>
<td>0.273</td>
<td>0.282</td>
</tr>
<tr>
<td>FT+ERINE</td>
<td>0.332</td>
<td>0.304</td>
</tr>
<tr>
<td>FakeBERT</td>
<td>0.352</td>
<td>0.334</td>
</tr>
<tr>
<td>ParallelBERT</td>
<td>0.363</td>
<td>0.332</td>
</tr>
<tr>
<td>PT+KPL</td>
<td>0.383</td>
<td>0.379</td>
</tr>
<tr>
<td>PT+SCL</td>
<td>0.398</td>
<td>0.397</td>
</tr>
<tr>
<td>PT+HGA</td>
<td><strong>0.426</strong></td>
<td><strong>0.417</strong></td>
</tr>
</tbody>
</table>

By contrast, our proposed Heterogeneous Graph Augmented Prompt-based Tuning (HGAPT) approaches demonstrated superior performance compared to their counterparts, achieving improvements ranging from 2.8% (3%) to 17.2% (17%) in terms of accuracy scores on the LiarFewShot (PolitifactFewShot) datasets with 6 classes. These results highlight the strong discriminatory power of our approach, particularly for low-resource fake news scenarios. Furthermore, when applied to the LiarFewShot (PolitifactFewShot) datasets with 2 classes, our method outperformed fine-tuning learning baselines, achieving an accuracy improvement of approximately 8% (11.3%). These findings suggest that metadata associated with news articles plays a critical role in distinguishing between fake and true news, and our approach effectively leverages this information to improve classification accuracy.

**D. Model Analysis**

To address EQ2, we conducted additional ablation studies on the various modules of HGAPT. Figure 4 and Figure 5 depict the experimental results obtained on the LiarFewShot dataset. In our evaluation, we first assessed the impact of constructing heterogeneous graphs in different ways. As shown in Figure 4, we observed that using only label information to construct the graph resulted in the worst performance. On the other hand, the performance of HGAPT improved as we incrementally added more metadata of news to construct the heterogeneous graph. This demonstrates the advantages of using more supervised information and constructing a heterogeneous graph, allowing us to capture specific topology among news.

We further explore different approaches to obtaining $\hat{e}_{ij}$ in equation 6. Specifically, we consider three methods: (i) $w_i/\hat{h}_{[CLS]}$, which sets $\hat{e}_{ij} = \cos(\hat{h}_i^{[CLS]}, \hat{h}_j^{[CLS]})$, where $\cos(\cdot, \cdot)$ denotes cosine similarity; (ii) $w_i/\hat{h}_{[MASK]}$, which sets $\hat{e}_{ij} = \cos(\hat{h}_i^{[MASK]}, \hat{h}_j^{[MASK]})$; and (iii) $w/\hat{y}$, which is the approach adopted in HGA and sets $\hat{e}_{ij} = \cos(\hat{y}_i, \hat{y}_j)$. Our experimental results, shown in Figure 5, demonstrate that HGA outperforms the other two methods. This validates that augmented class prediction carries more relevant information for discriminating news.

**E. Zero-Shot Fake News Detection**

Detecting fake news in minority domains has been shown to be difficult in previous studies [4], [8], [13] due to the lack of annotated resources. In order to address EQ3, we firstly compared various methods using different news samples, and

![Fig. 4. Effect of different heterogeneous graphs, where "l, p, t" devotes "Label, Publisher, Topic".](image)

![Fig. 5. Obtain edge representation in graphs, where "[CLS", "[MASK]" denotes "Label, different ways, Publisher, Topic".](image)

![Fig. 6. Effect of samples per class.](image)
evaluated their performance by measuring the accuracy on the LiarFewShot dataset obtained as we incrementally increased the number of samples. Figure 5 illustrates the impact of varying the number of labeled training samples on performance. It is evident that decreasing the number of training samples leads to reduce performance for all methods, with HGAPT consistently outperforming the other models.

Furthermore, we investigated the potential for low-resource fake news detection through domain transfer using HGAPT for zero-shot fake news detection. Specifically, these models were trained on a source training set and subsequently evaluated on the target test set. Table III presents the accuracy of several competitive models, highlighting the superiority of HGAPT over fine-tuning methods and state-of-the-art techniques. The results show that HGAPT achieves approximately 28% accuracy on the LiarFewShot dataset, and 25% accuracy on the PolitiFactFewShot dataset, which is substantially better than most of the baseline models. Taken together, these experimental findings suggest that HGAPT not only enhances detection performance, but also exhibits superior zero-shot fake news detection capabilities.

VI. CONCLUSION

Our work introduces HGAPT, a novel prompt-based tuning framework that leverages heterogeneous graph augmentation to address the challenge of detecting low-resource fake news on social media. During the learning process, HGAPT constructs batch-wise heterogeneous graphs based on the metadata associated with each news item, and utilizes this information to fine-tune pretrained language models for solving both fake news classification and fake news relation prediction problems. By leveraging the limited supervised information available for news items, HGAPT is able to fully exploit the available data and achieve significant improvements over state-of-the-art models on both fake news classification and zero-shot detection tasks. Extensive experiments were conducted on two real-world datasets to demonstrate the effectiveness of our proposed model.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China No. 62176187, the Major Projects of the National Social Science Foundation of China No.11ZD189.

REFERENCES


Abstract—It is challenging to synthesize the input grammars for complex parsing programs. To address this issue, this paper proposes a novel token-based synthesis method for learning input grammars. The key idea is to synthesize the input grammar at the token level, rather than the character level, which improves both the synthesis efficiency and the grammar’s completeness. Specifically, we propose using token-based symbolic execution to automatically generate valid token sequences. Then, we propose a token-level grammar synthesis method that incorporates a novel generalization operation to improve the generalization of the grammar. Additionally, we utilize SMT optimization to generalize the character representation of each token to enhance the grammar’s precision. The preliminary experimental results is promising.

Index Terms—Grammar, Synthesis, Token, Symbolic Execution

I. INTRODUCTION

Software inputs are typically expected to adhere to specific formats. If an input does not comply with the format requirements, the software will typically reject it and may generate input errors. For more intricate and structured input formats, there are often input grammars that can be expressed using regular expressions or context-free grammars, such as programming language grammars. Nonetheless, there may be situations where the input grammar is not readily available [1]. Input grammars are generally considered to be the domain of software developers. Even in cases where there are descriptions of the input grammar in natural language, such descriptions are often incomplete and rarely in a machine-readable format. Furthermore, manually creating an input grammar can be a time-consuming task [2]. When the input grammar of the software is complex and not available, the automatic testing of the software becomes significantly challenging.

State-of-the-art methods for grammar synthesis [3]–[5] generally presume that a set of initial valid inputs are available. These inputs are typically provided in character-level format, such as strings or byte-level files. However, the availability of valid inputs for a program can be limited, and it may be challenging to ensure that such inputs have a representative distribution that covers all production rules and terminals in the grammar. One promising approach to address this issue is to leverage symbolic execution to generate valid character-level inputs for grammar learning [1]. This approach involves using a set of sample strings to derive a series of generalizations, but the large search space associated with character-level methods may render the learning process inefficient and limit the effectiveness of the learned grammar.

Our key observation is that tokens play a critical role in grammar. The typical parsing process involves tokenizing the input into a sequence of tokens during lexical analysis, followed by verification of the token sequence based on syntax rules during syntax analysis. As token sequences are more closely aligned with the productions in the grammar, learning the grammar at the token level could potentially lead to more efficient and effective learning, resulting in a more precise and comprehensive learned grammar.

Based on this insight, we present an approach for synthesizing input grammars by leveraging token-level symbolic execution. Unlike the existing methods, our approach is a white-box learning approach that leverages grammar-agnostic symbolic execution [6] to generate token sequences for programs with complex input formats. These generated token sequences are then used to synthesize the corresponding input grammars. To enhance the generalization process, we introduce a new operation that enables the exchange of sub-expressions within regular expressions, thereby expanding the scope of generalization. Furthermore, we propose a novel method for generating token values that utilizes Satisfiability Modulo Theories (SMT) optimization to generalize the possible character-level values of a given token, thereby improving the precision of the synthesized grammar.

II. SYNTHESIS FRAMEWORK

Our synthesis framework is illustrated in Figure 1. It is composed of two distinct phases: the token learning phase and the grammar synthesis phase. The first phase is carried out by means of grammar-agnostic dynamic symbolic execution (GADSE) [6]. Specifically, GADSE analyzes a given program in two stages. In the first stage, it performs the dynamic symbolic execution of the program’s tokenization code, collecting the character-level constraints for each token value. In the second stage, GADSE symbolizes the generated tokens rather than each individual character of the input, producing the corresponding token-level path constraints. After executing the program concretely, GADSE generates a new token-level path constraint that corresponds to an alternative input grammar case, thereby enabling the generation of a new token sequence.

The second phase of our synthesis framework utilizes the token sequence and token constraints generated during the first phase to synthesize the input grammar of the program. The Grammar Synthesis phase comprises two distinct steps. Firstly, a token-level context-free grammar is learned through token-based grammar synthesis. Subsequently, the learned token-level grammar is generalized into a character-level grammar by means of token generalization.
A. Token-level Context-free Grammar Synthesis

1) Regular Expression Generalization: During the first phase of synthesis, the input string is generalized into a regular expression. In contrast to the generational operations employed in GLADE, we propose a novel operation called Exchange, which facilitates generalization by merging the first and third parts of the regular expression if they are interchangeable. This operation enhances GLADE’s generalization ability on single input. Moreover, Exchange ensures that the resulting generalized expression remains a valid regular expression and retains the structure of the expression prior to generalization. The generalization conforms to the following meta-grammar.

\[
T_{rep} ::= \beta \mid T_{alt}^* \mid \beta T_{alt}^* T_{rep} | T_{alt} T_{rep} \quad \text{(Repetition)}
\]

\[
T_{alt} ::= T_{rep} \mid T_{rep} + T_{alt} \quad \text{(Alternation)}
\]

\[
T_{exh} ::= (T_{meg} T_{rep})^* T_{meg} \quad \text{(Exchange)}
\]

\[
T_{meg} ::= T_{rep} + T_{rep}
\]

\(T_{rep}\) represents Kleene star, \(T_{alt}\) represents alternation, \(T_{exh}\) represents exchanging sub-expressions, and ranges over all substrings of the input string. Hence, each generalization step of regular expression can be translated into one or multiple rules in the context-free grammar.

In the context of our token-based synthesis method, each generalization setup may have multiple candidates due to the existence of multiple decompositions. However, this limitation is less severe than character-level generalization, since token sequences are typically shorter than character-level inputs. As a result, we only need to consider the decompositions between tokens, which are fewer than those between characters.

2) Context-free Grammar Generalization: The second phase of grammar synthesis involves the generalization of the regular expression obtained in the first phase into a context-free grammar. This phase comprises two steps. Firstly, the regular expression is translated into an equivalent context-free grammar by leveraging the production rules of the meta-grammar corresponding to the first stage’s generalization of the regular expressions. Subsequently, we generalize the context-free grammar by merging non-terminals derived from repetition and exchange operations, which may introduce recursions in the resulting grammar. Notably, in our approach, each token is assigned a dedicated non-terminal, and the non-terminals of tokens can also be merged during this stage, representing an advantage of synthesizing at the token-level.

B. Token Generalization

Since our approach synthesizes the grammar at the token-level, the resulting grammar is also a token-level context-free grammar. To fully complete the grammar, we need to generate the character-level values for each token. This is achieved by synthesizing a grammar for each token and utilizing the constraints collected by GADSE in the first stage of our synthesis method. As a token’s values can often be represented by regular expressions, we employ the first stage of our synthesis method to synthesize the seed values for each token. Furthermore, to refine the search space for token generalization, we use the token’s constraints to calculate the range of each character’s value through SMT optimization.

III. PRELIMINARY EVALUATION AND NEXT STEP

We have implemented our approach for Java programs using GLADE [3] and GADSE [6], with the underlying Satisfiability Modulo Theories (SMT) solver of GADSE being Z3. The effectiveness of synthesized grammars is evaluated through three metrics: precision, recall, and F1 score. The precision is determined by calculating the proportion of inputs sampled from the synthesized grammar that is accepted by the oracle grammar. On the other hand, the recall is determined by calculating the proportion of inputs sampled from the oracle grammar that is accepted by the synthesized grammar. Efficiency is measured by comparing the synthesis time of different synthesis methods. Our approach is evaluated using three real-world Java parsing programs\(^\d\) with complex and diverse input grammars. We compared our approach with two existing state-of-the-art methods: GLADE and Arvada [5]. Our approach achieves 17x and 19x higher F1 scores, respectively, on average. Furthermore, our approach achieves a high efficiency compared with both GLADE and Arvada.

The subsequent steps entail three key aspects: (1) The evaluation on extensive parsing programs; (2) The application of our approach in fuzzing or symbolic execution to enhance its efficiency; (3) The exploration of a synthesis method for context-sensitive grammars.

REFERENCES


\(^\d\)https://github.com/mwnorman/JSONParser

\(^2\)https://github.com/abcdw/javacc-clojure

\(^3\)https://github.com/mwnorman/JSONParser
Session CVA: computer Vision and Applications
YOLOv7-marine: An Improved YOLOv7 Model for Object Detection in Marine Environments

1st Puhui QU  
Key Laboratory of Underwater Acoustic Communication and Marine Information Technology  
Xiamen University  
Xiamen, China  
qupuhui@163.com

2nd Keyu CHEN*  
Key Laboratory of Underwater Acoustic Communication and Marine Information Technology  
Xiamen University  
Xiamen, China  
chenkeyu@xmu.edu.cn

3rd En CHENG  
Key Laboratory of Underwater Acoustic Communication and Marine Information Technology  
Xiamen University  
Xiamen, China  
chengen@xmu.edu.cn

Abstract—In this paper, we propose a novel target detection algorithm that addresses the challenge of difficult recognition and localization in sea surface general purpose target detection. The proposed algorithm is based on an improved YOLOv7, incorporating an efficient non-parametric attention mechanism module-SimAM into the original network, which reduces the model parameters and enhances the expressiveness of the network as well as the extraction ability of the model for important features. Additionally, we introduce a new module, CN-CSP, that merges the strengths of CSP and ConvNext, thereby improving the network’s learning ability while reducing the computational overhead. Furthermore, the integration of the rssp module into the backbone of YOLOv7 enables the network to extract features in a more comprehensive and multi-scale manner. Experimental results on The Sea Surface Target Dataset indicate the superiority of the proposed algorithm, achieving detection accuracy of 78.3% with improvements of 3.1% compared to the original YOLOv7 model.

Index Terms—Yolov7; ConvNext; marine target detection; attention mechanism;

I. INTRODUCTION

The ocean holds a significant position in the global economic growth, and its exploration and utilization of marine resources is vital. Recently, deep learning techniques have gained considerable attention and have been increasingly used in various real-world applications, including object detection, video surveillance, autonomous driving, and face recognition. These deep learning-based target detection algorithms demonstrate better results than traditional methods and are characterized by faster detection and higher accuracy. Target detection algorithms can be classified into two categories: two-stage and one-stage target detection algorithms. Two-stage algorithms such as SPP-Net [1], Faster R-CNN [2], and R-FCN [3], first generate the region proposal and then perform detection. Conversely, one-stage algorithms such as SSD [4] and YOLO [5]–[10] series algorithms directly obtain the location and class information of the target and exhibit faster detection. The major contributions of this work can be summarized as follows:

1. A CN-CSP architecture is proposed, which is derived from the combination of ConvNext [11] and CSPnet [12]. This architecture not only enhances the learning capability of CNN, but also significantly improves the target detection performance.

2. We have optimized the MPConv module in YOLOv7 [10] by incorporating the SimAM [13] module to create the Sim_MPConv module. The Sim_MPConv module effectively suppresses irrelevant information without adding any additional parameters, leading to improved performance without increasing computational complexity. It also enhances detection accuracy and recognition of small targets on the sea surface.

3. A novel RSPP module is introduced, which integrates the residual and SPP [1] structures. This module effectively extracts rich feature information from the input image, enabling a deeper network, and results in improved detection accuracy.

II. ARCHITECTURE

A. The Overview of the YOLOv7 algorithm

The YOLOv7 algorithm [10] optimizes the balance between detection accuracy and efficiency through innovative strategies. Specifically, it integrates the extended efficient long-range attention network (ELAN), model scaling using cascaded models [14], and convolutional reparameterization [15]. The YOLOv7 network comprises three modules: Backbone, Neck, and Prediction. The Backbone module contains ELAN and MPConv convolutional layers. The ELAN layer increases feature diversity by directing different feature groups to enhance learning ability without compromising gradient paths. The MPConv module has a convolutional layer and Maxpool layer, forming two branches. Their features are combined by Cat to improve feature extraction. The Neck module uses a Path Aggregation Feature Pyramid Network (PAFPN) [14] to fuse features from different levels through bottom-up paths, enabling smoother information transfer from lower to higher levels. The Prediction module adjusts the channel numbers for P3, P4, and P5 features from the PAFPN using RepVGG Blocks [15]. Finally, 1x1 convolution predicts confidence, category, and anchor boxes.

This work was supported in part by the National Natural Science Foundation of China (62071402 and 62271425).
B. Integration of Efficient 3D Attention Module

The attention mechanism plays an important role in facilitating the effective identification of key regions in complex visual scenes. As shown in Eqs(1) to (3) and the left part of Figure 1, SimAM [13] module evaluates each neuron in each network by defining a linear differentiability energy function, where \( t \) is the target neuron, \( x \) is the neighboring neuron, and \( \lambda \) is the hyperparameter. \( e \) lower energy indicates that the neuron is more differentiated from its neighbors, and the neuron is more important. The neurons are weighted according to importance by \( 1/E \) as shown in equation (4).

\[
e_i^* = \frac{4(\hat{\sigma}^2 + \lambda)}{(t - \hat{\mu})^2 + 2\hat{\sigma}^2 + 2\lambda} \quad (1)
\]

\[
\hat{\mu} = \frac{1}{M} \sum_{i=1}^{M} x_i \quad (2)
\]

\[
\hat{\sigma}^2 = \frac{1}{M} \sum_{i=1}^{M} (x_i - \hat{\mu})^2 \quad (3)
\]

\[
\tilde{X} = \text{sigmoid} \left( \frac{1}{E} \right) \odot X \quad (4)
\]

Fig. 1. The SimAM module and the Sim_MPConv structure.

In our experiments, we integrated the SimAM attention mechanism with MPConv to formulate the Sim_MPConv module, as illustrated in Figure 1 on the right. The Sim_MPConv module enhances the contribution of neurons that convey more relevant information and effectively mitigates the impact of irrelevant features, thereby strengthening the network’s feature representation capability and enhancing the model’s target localization accuracy while reducing the influence of background interference.

C. Efficient CN-CSP Structure

The ConvNext-Block structure originates from the ConvNext [11] architecture, and we propose a novel integration of the ConvNext-Block with the CSP structure to form the ConvNext-Block-CSP (CN-CSP), which is depicted in figure 2.

The CN-CSP structure employs the Layer Normalization (LN) layer, which stabilizes the model and reduces the oscillation of gradients during training. The structure incorporates the Gaussian Error Linear Unit (GELU) [16] function, which not only overcomes the gradient vanishing issue, but also accelerates the training speed compared to the traditional sigmoid function. This module effectively enhances the performance of the network while also optimizing the utilization of each computing unit and reducing extraneous resource consumption.

D. The RSPP Structure

The residual edges present in the residual [17] structure are pivotal in enabling the network to learn nonlinear representations and accelerate the training process. Moreover, the spatial pyramid pooling layer, as shown in figure 3 on the left part, effectively captures both global and spatial information of the feature map, significantly enhancing the network’s generalization ability. Through the integration of these two components and additional modifications, the Residual Spatial Pyramid Pooling (RSPP) structure is formed, as depicted in figure 3 on the right. The RSPP structure enables the extraction of features more efficiently, leading to improved prediction performance without sacrificing computational efficiency.

Fig. 2. The ConvNext-Block and the structure of CN-CSP.

E. The Yolov7-marine Architecture

As depicted in figure 4, the improved network architecture of YOLOv7 is presented. By incorporating the RSPP structure into the backbone of YOLOv7, the deep feature extraction capability of the network is improved. The MpConv structure in YOLOv7 is then advanced by adding the 3D attention mechanism, SimAM, to form Sim-MPConv, which enhances the feature extraction capability. Finally, by integrating the CN-CSP structure into the neck of YOLOv7, the learning ability of the CNN is boosted while making the network more efficient.

Fig. 3. The SPP Module and the RSPP structure.

III. EXPERIMENTS

In order to evaluate the performance of the improved YOLOv7 algorithm, we trained and evaluated the algorithm on a commonly used dataset of The Sea Surface Targets.

A. Experimental Setup

We use Common target dataset on The Sea Surface Target Dataset to conduct experiments and validate our object detection method. All our experiments did not use pre-trained models. That is, all models were trained from scratch.
Fig. 4. Improved network model based on Yolov7 network.

The Sea Surface Target Dataset, consisting of 7150 images, encompasses ten categorical classes, specifically lighthouse, sailboat, buoy, railbar, cargoship, navalvessels, passengership, dock, submarine, and fishingboat. The dataset is partitioned into training set, validation set, and test set, utilizing a ratio of 7:2:1, respectively.

In order to rigorously evaluate the performance of the proposed method, a series of experiments were carried out with the following parameter settings. The input images were preprocessed by resizing to a resolution of 640x640 pixels. The optimization algorithm employed in the experiments was Stochastic Gradient Descent (SGD), with a learning rate of 1e-2 applied to the model via a weight decay of 5e-4, and the learning rate was adapted via the Cosine Annealing schedule. The batch size for training was set to 20, with a total of 300 training epochs being performed. The software environment for the experiment is: operating system Linux, Python 3.10, PyTorch 1.11.0, CUDA 11.5.2, GPU: RTX 3090.

### Table I

<table>
<thead>
<tr>
<th>Method</th>
<th>RSPP</th>
<th>Sim_MPCov</th>
<th>CN-CSP</th>
<th>mAP0.5 (%)</th>
<th>Params (M)</th>
<th>FLOPS (G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>75.20</td>
<td>36.90</td>
<td>104.70</td>
</tr>
<tr>
<td>G2</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>76.30</td>
<td>36.90</td>
<td>104.70</td>
</tr>
<tr>
<td>G3</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>76.60</td>
<td>38.72</td>
<td>108.10</td>
</tr>
<tr>
<td>G4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>76.90</td>
<td>38.72</td>
<td>108.10</td>
</tr>
<tr>
<td>G5</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>77.10</td>
<td>37.58</td>
<td>107.00</td>
</tr>
<tr>
<td>G6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>78.30</td>
<td>39.05</td>
<td>109.80</td>
</tr>
</tbody>
</table>

### B. Ablation Experiments

In order to evaluate the contribution of the proposed improvements to the overall performance of the model, we conducted a set of ablation experiments on the Sea Surface Target Dataset. The results of these experiments are summarized in Table 1, which indicates the utilization of the RSPP, CN-CSP, and SimAM methods. These ablation experiments were able to evaluate each of the improvements and allowed us to assess their efficacy in improving model performance.

As illustrated in Table 1, the performance evaluation of the YOLOv7 algorithm was carried out in several experiments. The first experiment showed a detection accuracy of 75.2% with the original YOLOv7 algorithm. The second experiment aimed to enhance the detection accuracy by incorporating the SimAM module into the YOLOv7 algorithm. This integration resulted in an improvement of 1.1% in the detection accuracy, without increasing the number of model parameters. The third experiment focused on integrating the CN-CSP structure into the YOLOv7 algorithm, which resulted in an improvement of 1.4% in the detection accuracy, with a small increase in the computational cost of the model. The fourth experiment combined the structures from the second and third experiments, resulting in a further improvement of 1.7% in the detection accuracy. The fifth experiment added the RSPP structure to the YOLOv7 algorithm, which resulted in an improvement of 1.9% in the detection accuracy. The final experiment incorporated the RSPP structure into the fourth experiment, resulting in a substantial improvement of 3.1% in the detection accuracy.

In conclusion, these experiments demonstrate the effectiveness of incorporating different structures into the YOLOv7 algorithm, in improving its detection accuracy.

### C. Analysis

In this study, we conduct a comprehensive evaluation of the improved YOLOv7 algorithm in comparison to the current mainstream target detection algorithms on The Sea Surface Target Dataset. The experimental results are presented in Tables 2. The results reveal the superiority of the improved YOLOv7 algorithm in terms of accuracy compared to the existing methods.

As demonstrated in Table 2, the improved YOLOv7 model achieved an average accuracy of 78.3% on The Sea Surface Target Dataset, outperforming YOLOv5s (6.0) and YOLOx-s by 7.1% and 6.3%, respectively, in terms of accuracy. In addition, the improved YOLOv7 model showed a remarkable improvement in detection accuracy with regards to map0.5 and map0.5-0.95, with an increase of 3.1% and 1.5% over the original YOLOv7, respectively.

Figure 5 illustrates a comprehensive performance evaluation of various algorithms on The Sea Surface Target Dataset. The first row displays the detection of cargo ships under foggy conditions. The results reveal that the improved YOLOv7 algorithm has an accuracy of 87%, which is 5% higher than YOLOv7. The second row showcases the success of the improved YOLOv7 algorithm in identifying small targets on the sea surface that were previously missed by YOLOv7. Moreover, the improved YOLOv7 network demonstrates greater accuracy than YOLOv7 for identifying other targets. The third row demonstrates that the overall accuracy of the improved YOLOv7 model is higher than that of the YOLOv7 network under normal conditions.

### IV. Conclusion

In this paper, we address the challenges in surface target detection by improving the YOLOv7 network architecture. The proposed improvements aim to tackle the difficulties in detecting surface targets and coping with complex surface
environments. The network structure is enhanced by incorporating the MPConv module from the original architecture with an efficient 3D attention mechanism to form the Sim_MPConv structure, and by adding the CN-CSP structure to the YOLOv7 neck, which enhances feature extraction capabilities. Additionally, the RSPP structure is utilized in the YOLOv7 backbone to significantly improve network prediction accuracy. Experimental results demonstrate that the proposed method effectively improves the accuracy of sea surface target detection without significantly increasing the number of model parameters or computational effort. The effectiveness of the approach has been validated on both the Sea Surface Target Dataset indicating its general applicability and potential for practical use. Further improvement in accuracy for sea surface target detection is expected in future studies.

REFERENCES


Object-Level Contrast Learning for 3D Sparse Object Detection in Ocean Scene

Yuheng He, Wenbin Yang, Xiangfeng Luo*, Liyan Ma, Shaorong Xie

College of Computer Engineering and Science, Shanghai University
Shanghai, China

rianh@shu.edu.cn, youngwb@shu.edu.cn, *luoxf@shu.edu.cn, liyanma@shu.edu.cn srxie@shu.edu.cn

Abstract—LiDAR-based 3D object detection provides the necessary high-precision environmental sensing information for the safe navigation of smart ships. However, relying on viewpoint projections, voxelized point clouds, or using inefficient point sampling methods, current LiDAR 3D object detection methods treat all objects uniformly and quantitatively while ignoring the specificity of sparse objects in the scene, which leaves less useful information about sparse objects. In this paper, we propose an end-to-end two-stage architecture, Object-Level Contrast Learning 3D Object Detection network (OCL), for better construction of sparse object features and improving the ability of model to detect sparse objects. In the first stage, the Contrast Learning based Sparse Object Feature Enhancement training strategy is proposed to decrease the feature discrepancy between sparse and regular objects in object-level. In the second stage, we use the Point-level Feature Multiple Aggregation Strategy to aggregate finer point-level features of sparse objects. Extensive experiments show that OCL achieves excellent performance on both Ship dataset and KITTI dataset. Furthermore, our work proposes a promising new idea for applying contrast learning to 3D object detection.

I. INTRODUCTION

Accurate 3D object detection is critical to ensure safe navigation of smart ships, since the precise localization information it provides directly affects the effectiveness of downstream tasks. LiDAR, as a high-precision detection instrument, offers more precise position information for objects than conventional images, and the LiDAR-based object detection keeps a promising research area with significant potential.

However, there is a high variability in the sparsity of LiDAR data due to the distance, the factors in LiDAR itself and the environment, as shown in Figure 1. Objects closer to the sensor are denser, while more distant objects are sparser. Moreover, high-powered LiDAR used for ocean navigation may also overheat, resulting in sparse objects. As a consequence, objects with more points and distinct contours are more easily detected, while those with fewer points and weaker contours are more challenging to detect. The topic of how to efficiently identify sparse objects remains a challenging research subject.

The methods for 3D object detection have been extensively studied from various perspectives, including 2D projection-based methods [1], [2], 3D voxel-based methods [3]–[6], and point-based methods [7]–[10]. These methods quantify all objects uniformly and ignore the specificity of sparse objects in the scene, which makes sparse objects have even less useful information. In an effort to generate finer feature representations of objects, some researchers obtain voxel-level features and then aggregates more precise point-level features [11]–[15]. This kind of framework has been very popular in recent studies, showing strong performance across a variety of scenarios, and presenting opportunities for further advancements, particularly in the detection of sparse objects.

In this paper, We propose Object-Level Contrast Learning 3D Object Detection network (OCL), an end-to-end two-stage detector that addresses the unique feature specificity of sparse objects in LiDAR data. OCL consists of two crucial modules: Contrast Learning based Sparse Object Feature Enhancement (CLFE) and Point-level Feature Multiple Aggregation (PFMA). To achieve better voxel-level feature representation of sparse objects, we propose CLFE in the first stage, which decreases the feature discrepancy between sparse and regular objects by contrast learning. The feature of sparse objects keeps more abstract than regular objects. Therefore, in the second stage, we propose a finner feature aggregation method PFMA to better perform point-level feature aggregation on sparse objects.

The effectiveness of our proposed theories has been demonstrated through sufficient experimentation. Notably, the OCL has proven to be highly effective for detecting sparse objects in both the Ship dataset and the KITTI dataset, surpassing the performance of existing methods. There is a 2.04% and 1.77% improvement in the mAP metric on the Ship dataset and KITTI dataset, respectively, compared to the current methods. Furthermore, our work proposes a promising new idea for applying contrast learning to 3D object detection. We summarize the following contributions of our method:

- By using contrast learning, we propose a sparse object feature enhancement strategy to decrease the voxel-level feature discrepancy between sparse and regular objects.
- Building upon prior research, we propose a multiple aggregation strategy for finner point-level feature aggre-
gation of sparse object.

- Our proposed method shows excellent performance on both Ship dataset and KITTI dataset compared to the current methods.

II. RELATED WORK

3D Object Detection. Voxel-based methods [3]–[6] represent point cloud quantitatively with voxels for rapid analysis of disordered point clouds using 3D Convolution. Point-based methods [7]–[10] usually sample key points first to reduce computation, and then use a symmetric function to extract point-level features. Point-voxel-based methods [11]–[15] use both voxel-level and point-level features of the point cloud to combine their advantages for better detection. SA-SSD [11] proposes an auxiliary network, which interpolates point-level features to intermediate voxel layers, to auxiliary supervise the training of backbone network. PV-RCNN [12] and PVR-CNN++ [13] use a two-stage structure to get more precise bounding box. In the first stage, getting the rough estimated ROI region through the voxel-level backbone network and RPN. In the second stage, aggregating the point-level feature for precise bounding box generation. PDV [15] uses the correlation between density and distance in point cloud to aggregates more information in the second stage. Point-Voxel-based methods are a popular research trend and have demonstrated strong performance across various scenes, particularly in larger-scale scenes. However, these methods still have great potential for detecting sparse objects.

Contrastive learning. Contrast learning [16] is a popular methodology that has been used in various fields recently [17]–[21]. The basic concept of contrast learning is to create a metric space where similar pairs of samples are pushed closer together and dissimilar pairs are pushed further apart. The most prevalent approach for implementing contrastive learning is to apply some global transformations to the image and then use a siamese network [20], [21]. The siamese network can be trained to learn differences and similarities between pairs of inputs. This approach has proven to be simple and effective for downstream tasks such as classification and regression that rely on global features. However, object-level features are more important for 3D object detection task than global features of scene. How to effectively use contrast learning method in 3D object detection task is still an area that needs to be explored.

III. METHOD

A. Network Architecture

Our network designed as a two-stage detector, as shown in Figure 2. In the first stage, we utilize the backbone network, refer to Second [4], with both 3D sparse convolution layers and 2D convolution layers to generate ROI regions from the point cloud. The Contrast Learning based Sparse Object Feature Enhancement module is also built on the feature extraction part, which will be detailed in section III-B. In the second stage, we use the Point-level Feature Multiple Aggregation (PFMA) module to create more precise local features for sparse objects, which will be detailed in section III-C. Besides, we also fuse BEV-level ROI features to add global features for objects. Finally, the parameters associated with the bounding box of object will be optimized by classification and regression.

B. Contrast Learning based Sparse Object Feature Enhancement

There is a significant difference in the distribution of points between sparse and regular objects, which results in a large semantic gap in their feature representation on the feature map and limits the detection performance of sparse objects. To address this issue, we propose the CLFE module, which is specifically designed in object-level to decrease the feature discrepancy between sparse and regular objects. This module is only activated during training and does not add any time cost burden to inference. Here we call the points in ground-truth(GT) bounding box as info-points, and the features used to judge the class of objects as intrinsic-feature. We perform object intrinsic-feature invariant transformations on the info-points of all objects in the input point cloud, such as local down-sampling, center rotation and scaling, to simulate the points distribution of sparse objects in the scene. The generated point cloud scene is input to the backbone network together with the raw point cloud scene as a pair. Due to the characteristics of the organization of the voxel space, we can derive the position of each object on the feature map from the step information in the convolution process. As mentioned above, the corresponding position \( \mathcal{P} \) and size \( S \) of each GT on the feature map can be formulated as:

\[
\mathcal{P}_i = \left( \frac{GT_i^x}{\text{stride}_x}, \frac{GT_i^y}{\text{stride}_y} \right)
\]

(1)

\[
S_i = \left( \frac{GT_i^x}{\text{stride}_x}, \frac{GT_i^w}{\text{stride}_y} \right)
\]

(2)

where \( GT_i^x, GT_i^y, GT_i^w \) denote the coordinates along X-axis, Y-axis, and length, width of the ith GT, respectively; \( \text{stride}_x, \text{stride}_y \) denote the spatial scaling of the current feature map relative to the original point cloud space.

For computational convenience, we take the smallest square region containing the ith object on the feature map as the contrast loss denoted as \( \Gamma_i \in R^{\max(S_i) \times \max(S_i) \times C} \). The ith object feature region in the original scene and the transformed scene are denoted as \( \Gamma_i^O \) and \( \Gamma_i^T \). Conventional contrast methods, which directly calculating the feature gap between \( \Gamma_i^O \) and \( \Gamma_i^T \), may cause the network to fall into extreme local optimization, resulting in the network not learning any useful information. Inspired by SimSiam [21], We apply a linear projection \( \varsigma \) to a portion of the feature pair to enable the training of the network to proceed as we expect, and the contrast loss \( L_{\text{contrast}} \) can be formulated as:

\[
L_{\text{contrast}} = C(\Gamma_i^O, \varsigma(\Gamma_i^T))
\]

(3)

where \( C \) denotes the calculation method of feature distance.

The feature region needs to be transformed into feature vectors before it is used for contrast loss, and we propose several feature transformation methods: direct contrast mode, average value mode, maximum activation mode, and mean-maximum activation mode, which we will discuss specifically in the
ablation experiments. The total loss of this part \( L_{\text{auxiliary}} \) can be formulated as:

\[
L_{\text{auxiliary}} = \sum_{i}^{N_{\text{GT}}} (W^{O} \times D^{O}_{i} + W^{T} \times D^{T}_{i})
\]

\[
D^{O}_{i} = 1 - C(\gamma(\Gamma^{O}_{i}), \gamma(\varsigma(\Gamma^{O}_{i})))
\]

\[
D^{T}_{i} = 1 - C(\gamma(\Gamma^{T}_{i}), \gamma(\varsigma(\Gamma^{T}_{i})))
\]

where \( W^{O}, W^{T} \) is the predefined loss weight corresponding to original and transformed objects, \( D^{O}, D^{T} \) is the calculated feature distance of original and transformed objects, \( \gamma \) is the feature transformation method, and \( N_{\text{GT}} \) denotes the number of GT in the current scene. We choose cosine similarity function as the distance calculation method \( C \).

C. Point-level Feature Multiple Aggregation

Numerous studies have investigated ROI feature aggregation method. PV-RCNN [12] uses the Farthest Point Sampling (FPS) [22] to globally sample key points for feature aggregation. PDV [15] directly attaches voxel features to the gravity center of voxels (GCV) and uses them as key points for subsequent feature aggregation to reduce the complexity of FPS sampling and original point feature extraction. However, these methods reduce the effective information of sparse objects in the process of sampling or quantizing. Based on the existing works, we propose some finer ROI feature aggregation strategies aimed at improving the point-level feature representation of sparse objects.

Specifically, we propagate the voxel-level features obtained in the previous stage to the raw points within each voxel. The point-level features are then aggregated at a fine-grained level for the original points to form the local features of the object using the following strategy:

**Vanilla Aggregation.** For raw points that have been assigned voxel-level features, we operate on them directly through grid-based feature aggregation used in PDV [15].

**Voxel Gravity Points Aggregation.** First, The raw points with voxel-level features attached are interpolated to the GCV using a method similar to that used in PV-RCNN [12]. Then, the features on the GCV are aggregated to the ROI by Vanilla Aggregation. In this way, the feature offset problem caused by attaching voxel features directly to the GCV can be well corrected, and the features on GCV will contain more detailed information.

**Voxel Gravity Grid Points Aggregation.** First, the raw points with voxel-level features attached are aggregated to the GCV by Vanilla Aggregation. Then, the features of the GCV are aggregated to the ROI by Vanilla Aggregation again. The ROI features obtained by this two-step aggregation strategy will contain more detailed and deeper information about the semantic features of the objects.

The effectiveness of the above strategies will be verified...
in detail in the ablation study. Meanwhile, we impose a simple graph convolution module for enhancing the feature correlation between GCV to mine more sparse object features.

D. Training Losses

We use an end-to-end training strategy. The total loss $L_{total}$ consists of three parts, which are the contrast learning auxiliary loss $L_{auxiliary}$ for the feature extraction part, the region proposal loss $L_{RPN}$ for the first stage, and the suggested optimization loss $L_{refine}$ for the second stage:

$$L_{total} = L_{auxiliary} + L_{RPN} + L_{refine} \quad (7)$$

Where $L_{auxiliary}$ has been explained in detail in section III-B, the region proposal loss $L_{RPN}$ is composed of classification loss and box regression loss. The classification loss here adopts focal loss. The box regression loss adopts smooth-L1 loss. Therefore, the region proposal loss can be formulated as:

$$L_{RPN} = L_{focal} + L_{smooth-L1} \quad (8)$$

The proposal refinement loss $L_{refine}$ is also composed of classification loss and residual box regression loss. The classification loss here adopts IoU loss as same as PV-RCNN [12]. The residual box regression loss also adopts smooth-L1 loss. Thus, the proposal refinement loss can be formulated as:

$$L_{refine} = L_{IoU} + L_{smooth-L1} \quad (9)$$

IV. EXPERIMENTS

A. Datasets and Implementation Details

Datasets. We evaluate our model on both Ship dataset and KITTI [23] dataset. The Ship dataset is composed of actual ship data obtained from various locations such as ports and shoreside, utilizing 32-line and 128-line LiDAR. The LiDAR data is filtered, labeled, and divided into a training set (3427 samples) and a test set (856 samples) with a ratio of 8:2. The training set is used to train the detection model, while the test set is used to verify the effectiveness of our model. The 3D autonomous driving dataset KITTI also be divided into a training set (3712 samples) and a validation set (3769 samples) to further verify our theory. For the Ship dataset, the detection range is [-400m, 400m] for the X axis, [-50m, 450m] for the Y axis, and [-10m, 20m] for the Z axis. We divide the raw point cloud into voxels of size (0.5 m, 0.36m, 0.75m). For the KITTI dataset, the detection range is set to be [0, 70.4m] for the X axis, [-40m, 40m] for the Y axis, and [-3m, 1m] for the Z axis. We set the voxel size to be (0.05m, 0.05m, 0.1m).

Training and Inference Details. For the second stage point-level feature aggregation, we use the last two intermediate voxel-level features, and the spherical query radius is set to [3,6] in Ship dataset, [0.4,0.6] in KITTI dataset. For generating transformed point cloud data, we set the object scaling to [0.95,1.05], the object points down-sampling range to [0.1,0.4], and the minimum number of the object points threshold to 10. Our model is trained using Adam [24] optimizer with initial learning rate set to 0.01 and one-cycle strategy [25] for learning rate update. The training environment is RTX 3090 for a total of 80 epochs. For the preprocessing of the training data, we apply some 3D object detection data augmentation strategies, including global rotation, random flip, global scaling, and ground truth data augmentation [4].

B. 3D Detection on the Ship Dataset

Table I presents the performance of our proposed method on the Ship test set, where our method achieves optimal performance on CargoShip, EngineeringShip, and all categories. Specifically, the results on $AP_{R40}$ have shown a significant improvement of 2.92%, 2.41%, and 2.04% for CargoShip, EngineeringShip, and all categories, respectively, when compared to the current optimal results. In comparison to our benchmark method PDV, our proposed method outperforms PDV by 2.92%, 2.82%, 2.41%, and 2.04% on CargoShip, TourBoat, EngineeringShip, and mAP, respectively. Figure 3 intuitively shows the detection results of our proposed method and the benchmark model PDV on the Ship test set. The lidar in the scene is located in the lower center of the image. Our method accurately detects sparse objects, which effectively demonstrates the effectiveness of our method.

![Fig. 3. Snapshots of our 3D detection results on the Ship test set. Green boxes for CargoShip and yellow for ContainerShip.](image)

C. 3D Detection on the KITTI Dataset

We also evaluated the effectiveness of our proposed method on the KITTI validation set, and the results are shown in Table II. Our method achieves state-of-the-art multiclass results, with 3D $AP_{R40}$ improving 0.07%, 1.98%, and 2.47% on the moderately difficult car, pedestrian, cyclist categories, respectively, and a 1.77% improvement on the average multiclass accuracy mAP. Figure 4 visualizes the detection results of our proposed method and the benchmark model PDV on the KITTI validation set. Compared with the benchmark model, our method is also able to accurately detect some sparse objects at longer distances in the scene, which have fewer points than the sparse ship objects. This result further demonstrates the effectiveness of our method for detecting sparse objects.

D. Ablation Studies

We performed ablation experiments on the Ship dataset for each of our modules and validated the effectiveness of our modules using mAP.
TABLE I
THE SHIP TEST SET FOR MULTI-CLASS DETECTION, WITH 3D AVERAGE PRECISION OF 40 SAMPLING RECALL POINTS AND 0.7 INTERSECTION OVER UNION.

<table>
<thead>
<tr>
<th>Method</th>
<th>Car</th>
<th>Container Ship</th>
<th>Tour Boat</th>
<th>Engineering Ship</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second(2019)</td>
<td>51.52</td>
<td>98.89</td>
<td>89.47</td>
<td>55.13</td>
<td>73.75</td>
</tr>
<tr>
<td>PV-RCNN(2020)</td>
<td>82.88</td>
<td>100.00</td>
<td>96.71</td>
<td>84.18</td>
<td>90.94</td>
</tr>
<tr>
<td>IA-SSD(2022)</td>
<td>64.67</td>
<td>100.00</td>
<td>96.78</td>
<td>81.96</td>
<td>85.85</td>
</tr>
<tr>
<td>PDV(2022)</td>
<td>83.18</td>
<td>100.00</td>
<td>91.50</td>
<td>90.00</td>
<td>91.17</td>
</tr>
<tr>
<td>Ours</td>
<td>86.10</td>
<td>100.00</td>
<td>94.32</td>
<td>92.41</td>
<td>93.21</td>
</tr>
</tbody>
</table>

Improvement 2.92 2.41 2.04

TABLE II
THE KITTI VAL SET FOR MULTI-CLASS DETECTION, WITH 3D AVERAGE PRECISION OF 40 SAMPLING RECALL POINTS.

<table>
<thead>
<tr>
<th>Method</th>
<th>AP3D@Car-R40 (IoU=0.7)</th>
<th>AP3D@Pedestrian-R40 (IoU = 0.5)</th>
<th>AP3D@Cyclist-R40 (IoU = 0.5)</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy</td>
<td>Moderate</td>
<td>Hard</td>
<td>Easy</td>
<td>Moderate</td>
</tr>
<tr>
<td>VoxNet(2018)</td>
<td>81.97</td>
<td>65.46</td>
<td>62.85</td>
<td>57.86</td>
</tr>
<tr>
<td>Second(2019)</td>
<td>90.71</td>
<td>81.73</td>
<td>78.79</td>
<td>57.46</td>
</tr>
<tr>
<td>PV-RCNN(2020)</td>
<td>92.04</td>
<td>84.06</td>
<td>81.99</td>
<td>65.39</td>
</tr>
<tr>
<td>IA-SSD(2022)</td>
<td>90.94</td>
<td>83.00</td>
<td>80.04</td>
<td>56.98</td>
</tr>
<tr>
<td>PDV(2022)</td>
<td>92.39</td>
<td>85.11</td>
<td>82.78</td>
<td>64.77</td>
</tr>
<tr>
<td>Ours</td>
<td>92.76</td>
<td>85.18</td>
<td>83.00</td>
<td>66.70</td>
</tr>
</tbody>
</table>

Improvement 0.37 0.07 0.22

Components Ablation. We first conducted ablation experiments for each module in our network. As show in Table III, CLFE represent the Contrast Learning based Sparse Object Feature Enhancement strategy, PFMA represent the Point-level Feature Multiple Aggregation strategy. Exp.1 is the benchmark model PDV, while Exp.2 adds the CLFE strategy, resulting in a 1.88% improvement on mAP compared to the benchmark model, which validates the effectiveness of the CLFE training strategy. Exp.3 uses the PFMA as the ROI feature aggregation strategy, resulting in a 0.64% improvement on mAP compared to the benchmark model, which validates the effectiveness of the PFMA strategy. Experiment 4 is our proposed method, which improves mAP by 2.04% compared to the benchmark model.

TABLE III
ABLATION EXPERIMENTS OF TWO STRATEGIES ON NETWORK PERFORMANCE.

<table>
<thead>
<tr>
<th>Exp.</th>
<th>CLFE</th>
<th>PFMA</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>91.17</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td></td>
<td>93.05</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>✓</td>
<td>91.81</td>
</tr>
<tr>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>93.21</td>
</tr>
</tbody>
</table>

Methods For Measuring Feature Discrepancy Between Objects. In this part, we conducted ablation experiments on the methods that can be used to measure feature discrepancy between objects. In Table IV, Flatten indicates that the values of the two feature regions are flattened directly, and the flattened vector is used to calculate the contrast loss; Avg and Max indicates that we use an extra average-pooling layer and max-pooling layer, respectively, to further abstract the object features and compress them into a vector, which is then used to calculate the contrast loss; Avg-Max indicates that the vector used to calculate the contrast loss is obtained by concatenating the abstract object vector used in the Avg and Max strategies. As shown in Table IV, Exp.2 and Exp.3 are improved by 1.31% and 1.83%, respectively, compared with Exp.1, which indicates that the object feature vector after further abstraction by the pooling layer can better measure the feature discrepancy between sparse and regular objects, among which the max-pooling can relatively better express the object features. On the contrary, the performance of Exp.4 on mAP are decreased by 0.13% compared with Exp.3, which indicates that the object feature representations obtained from the two pooling layers mentioned above have some conflicts that cannot be used directly.

FIG. 4. Snapshots of our 3D detection results on the KITTI val set. Green boxes for car, cyan for pedestrian and yellow for cyclist.

Point-level Feature Multiple Aggregation Strategy. In this part, we validate the effectiveness of three different ROI
feature aggregation strategy, which have detailed in section III-C. As shown in Table V, VA represent the Vanilla Aggregation Strategy, VGPA represent the Voxel Gravity Points Aggregation Strategy, VGGPA represent the Voxel Gravity Grid Points Aggregation Strategy. The experimental results have 0.69% and 0.58% improvement on mAP for VGPA and VGGPA respectively, which well validate our previously proposed theory.

<table>
<thead>
<tr>
<th>Exp</th>
<th>VA</th>
<th>VGPA</th>
<th>VGGPA</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>90.54</td>
</tr>
<tr>
<td>2</td>
<td>√</td>
<td></td>
<td></td>
<td>91.23</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>√</td>
<td></td>
<td>91.81</td>
</tr>
</tbody>
</table>

V. CONCLUSION

Current methods treat all objects uniformly, ignoring the specificity of sparse objects in the scene, which leaves less useful information about sparse objects and is not conducive to the detection of 3D sparse objects in ocean scene. To address this limitation, Object-Level Contrast Learning 3D Object Detection network (OCL) is proposed, which is an end-to-end two-stage architecture that takes into account the feature specificity of sparse objects. In the first stage, the Contrast Learning based Sparse Object Feature Enhancement training strategy is designed in object-level to decrease the feature discrepancy between sparse and regular objects. In this procedure, the voxel-level features of sparse object can be enhanced. In the second stage, the Point-level Feature Multiple Aggregation strategy is utilized to better aggregate the point-level features of sparse object. The effectiveness of our proposed strategies are verified in extensive experiments on Ship dataset and KITTI dataset. Our work uses the characteristics of LiDAR data to generate sample pairs in a relatively simple way for contrast learning. However, the relationships between more objects in the scene have not been fully explored. Furthermore, we believe that there keeps a great deal of opportunity to explore the application of contrast learning in 3D object detection.

ACKNOWLEDGEMENT

This work is supported by the Development Project of Ship Situational Intelligent Awareness System under the Grant MC-201920-X01, the National Natural Science Foundation of China under the Grant No.61991415.

REFERENCES


Few-Shot Object Detection via Instance-wise and Prototypical Contrastive Learning

Qiaoning Lei, Yinsai Guo, Liyan Ma, Xiangfeng Luo*
School of Computer Engineering and Science, Shanghai University
Shanghai, China
Email: luoxf@shu.edu.cn

Abstract—Few-shot object detection (FSOD), which involves training the detector with few annotated data to detect novel objects, has aroused a wide range of research interests. However, the performance of FSOD is still limited by insufficient data. Existing works usually adopt fine-tuning paradigm, which first uses rich base classes for pre-training and then uses them to carve the novel class feature space. In the fine-tuning phase, the balance space learned by the pre-trained model will be broken leading to an intersection between the feature space of novel and base classes, which makes it difficult to distinguish the difference between them. Contrastive learning has been shown to learn a balanced feature space and enhance the discriminability of the learned features. Here, we present Few-Shot object detection via Instance-wise and Prototype Contrastive Learning (FS-IPCL), which introduces contrastive learning to learn a balanced feature space. FS-IPCL uses instance-wise and prototype contrastive loss during feature learning to enhance the intra-class compactness and inter-class separability of samples. In this way, the base and novel classes can be evenly distributed in the feature space, improving the class boundary to alleviate the confusion problem of the novel classes. Extensive experimental results on the PASCAL VOC and MS-COCO datasets demonstrate the effectiveness of the proposed method and achieve state-of-the-art performance.

Index Terms—Few-shot object detection; Contrastive learning; Graph convolutional network.

I. INTRODUCTION

With the widespread use of Deep Convolutional Neural Networks (DCNN) [1], [2] in recent years, object detection [3] algorithms based on DCNN have also advanced significantly. To assure the effectiveness of its detectors, object detection nevertheless depends on a sizable volume of annotated data. However, it requires considerable labor and time to get enough annotated data in the real world. Deep detectors are prone to overfitting when trained with only a few data, and their detection accuracy cannot be compared with that of detectors with a large amount of data. As opposed to that a child may quickly pick up novel categories and visual concepts by using a few examples. Therefore, most researchers are working to close this gap to give machines good perception capabilities.

Few-shot learning follows the principle of picking up new ideas rapidly, which can achieve promising performance with limited data. As a branch of few-shot learning, few-shot object detection (FSOD) is a more complex task compared to few-shot classification because it requires to locate the objects additional. The majority of FSOD approaches follow the meta-learning paradigm. The two-stage fine-tuning method (TFA) [4] has recently demonstrated more potential for improving FSOD because of its efficiency and simplicity compared to meta-learning techniques. To address the issue of sparse scale distribution of objects in FSOD, MPSR [5] proposes a multi-scale positive sample refining method on the basis of TFA. Nevertheless, the application difficulty of the method is increased due to the requirement for manual selection in its forward refinement branch. In this work, we find that fine-tuning using novel class samples is hard to learn a balanced discriminative feature space, making it difficult for the model to distinguish the features of the base classes and the novel classes.

Contrastive learning has been shown to learn a balanced feature space [6]. Since the distances and differences between categories are more obvious, contrastive learning can enhance the discriminability of the learned features. FSCE [7] uses supervised batch contrastive learning [8] to simulate the instance-level similarity and inter-class distinction of object proposal embeddings. It produces a more balanced feature space by optimizing the instance contrast loss, which can separate all instances well. However, since the model can distinguish different instances using low-level image differences, the learned feature embeddings may not capture the semantic knowledge of the objects. Inspired by FSCE, we introduce contrastive learning into FSOD with prototype contrastive learning. Prototypes are described as “representative embeddings of a class of semantically similar instances” [9]. It aggregates instances of the same class to obtain a compact image representation, which can capture some basic semantic structure of a single class.

We propose Few-Shot object detection via Instance-wise and Prototypical Contrastive Learning (FS-IPCL). Specifically, the instance-wise contrastive loss (ICL) improves the similarity of instances from the same class, and the prototype contrastive loss (PCL) improves the similarity between an instance and its corresponding prototype, so that the model learns more class-related semantic knowledge. However, two problems arise when contrastive learning is applied to FSOD: (1) there may be background noise in contrast samples; (2) there may exist positive and negative sample coupling [10]. That is, low quality positive samples will decrease the gradient

DOI reference number: 10.18293/SEKE2023-129
of a batch of informative negative samples and vice versa. To address these two problems, we screen contrast samples to obtain high-quality positive and negative samples and decoupling them. Additionally, the prototype constructed using the feature extracted in the initial stage may not be very reliable, and the feature representation of the image will constantly change during training. To obtain a representative prototype embedding, we adopt graph convolutional network (GCN) [11] to dynamically update the prototypes. Our approach has three main contributions:

- The instance and prototype contrastive loss are designed to learn a balanced discriminative feature space, improving by selecting high-quality positive and negative samples and decoupling them.
- To get more accurate prototype feature embeddings, GCN is used to dynamically update the prototypes in prototypical contrastive loss.
- A significant number of experiments are conducted by our method on the Pascal VOC and MS-COCO datasets, and new state-of-the-art results are achieved.

II. RELATION WORK

Few-Shot Object Detection. Currently, the commonly used FSOD methods are mainly based on meta-learning and fine-tuning paradigms. Following the meta learning methods, FSRW [12] proposes a reweighting module to extract the global features of the support image. For fine-tuning paradigm, TFA [4] first applies the two-stage fine-tuning method to FSOD, and proposes a new few-shot evaluation method. DeFRCN [13] uses a decoupling approach to solve the multi-stage and multi-task conflict problem when the Faster R-CNN detection framework is applied to FSOD. Our approach is also based on a two-stage fine-tuning paradigm.

Contrastive learning. Contrastive learning uses contrast to bring together similar classes and distinguish classes by constructing pairs of positive and negative samples. Most contrastive learning methods map features to unit hyperspheres for representation learning. A direct matching of uniformly sampled points on the unit hypersphere can provide a good representation. Existing contrastive learning is divided into two main categories, instance-wise contrastive learning [6] and prototypical contrastive learning [9]. Instance-wise contrastive learning brings similar instances close together and different instances far apart by comparing instance information from different samples. Prototypical contrastive learning represents the semantic structure of a class by using prototypes of image clusters. Some methods [14] have also used both instance-wise contrast learning and prototypical contrast learning. In this paper, contrastive learning is introduced into FSOD by mapping features onto unit hypersphere and instance-wise contrast learning and prototype contrast learning.

III. METHOD

Our proposed method FS-IPCL uses two-stage training. The base classes training step uses a rich base-class dataset to train the model. During the fine-tuning phase, we fine-tune the model using a relatively balanced small amount of base-class data and novel classes data (N-way K-shot). And the backbone feature extractor is frozen while the rest of the structure is fine-tuned. Meanwhile, we introduce ICL and PCL to supervise the RoI feature extractor, and jointly optimize the contrastive loss and the original classification and regression objectives. Our method’s overall architecture is shown in Figure 1.

A. Problem Setting

Our problem setting for FSOD follows the standard problem setting of previous works [4], [7], [13]. Our dataset is divided into a base set \( D_{\text{base}} \) with rich annotated instances, and a novel support set \( D_{\text{novel}} \) with only a few annotated instances per category. The class \( C_{\text{base}} \) in our base set \( D_{\text{base}} \) and the class \( C_{\text{novel}} \) in the novel support set \( D_{\text{novel}} \) do not overlap, that is, \( C_{\text{base}} \cap C_{\text{novel}} = \emptyset \). Our goal is to learn a robust detector which can recognize and localize the query set \( D_{\text{query}} \) pairs without annotated instances, where the class \( C_{\text{query}} \subseteq C_{\text{base}} \cup C_{\text{novel}} \) in \( D_{\text{query}} \).

B. Instance-wise and Prototypical Contrastive Loss

Since generic detectors struggle to capture discriminative region proposal features from a limited number of shots, we propose two contrastive losses to better distinguish feature representations of the novel classes: (1) ICL reduces similarity between object candidates from different classes and increases similarity between object candidates within the same category by comparing various RoI features. (2) PCL compares the RoI feature of the object with the prototype, making the object candidate box close to its corresponding class prototype and far away from other class prototypes. PCL first acquires the initial prototype \( \mathcal{C}' \in \mathbb{R}^{d_x \times K} \) and then utilizes the prototype updating operation to dynamically update the prototype. The prototype updates part which is described in III-C.

Inspired by supervised batch contrastive learning method [8], our ICL and PCL are designed as follows with suitable for FSOD. Specifically, we adopt the selected \( N^+ \) positive samples and \( N^- \) negative samples to reduce the noise influence in the samples. And we remove the similarity calculation of positive samples in the denominator to alleviate the influence of the coupling of positive and negative samples.

\[
L_{\text{ICL}} = -\frac{1}{N^+} \sum_{i=1}^{N^+} \frac{1}{N_{yi} - 1} \sum_{j=1, j \neq i}^{N^+} \frac{\exp \left( \sim z_i \cdot \sim z_j / \tau \right)}{\sum_{k=1}^{N^+ + N^-} \frac{\exp \left( \sim z_i \cdot \sim z_k / \tau \right)}{1 \{ y_i = y_j \} \cdot \log \sum_{k=1}^{N^+ + N^-} \frac{\exp \left( \sim z_i \cdot \sim z_k / \tau \right)}{1 \{ y_i \neq y_k \} \cdot \exp \left( \sim z_i \cdot \sim z_k / \tau \right)}} \tag{1}
\]

\[
L_{\text{PCL}} = -\frac{1}{N^+} \sum_{i=1}^{N^+} \frac{1}{N_{yi} - 1} \sum_{j=1}^{K} \frac{\exp \left( \sim z_i \cdot c_j / \tau \right)}{\sum_{k=1}^{K} \frac{\exp \left( \sim z_i \cdot c_k / \tau \right)}{1 \{ y_i = y_j \} \cdot \log \sum_{k=1}^{K} \frac{\exp \left( \sim z_i \cdot c_k / \tau \right)}{1 \{ y_i \neq y_k \} \cdot \exp \left( \sim z_i \cdot c_k / \tau \right)}} \tag{2}
\]
Fig. 1. Overview of our proposed FS-IPCL. The RPN and RoI feature extractors are fine-tuned in our approach in addition to the bounding box classifier and bounding box regressor. To introduce contrast learning, we select the RoI features for ICL, extracting features for the ground truth of all support images to compute prototype vectors for each class, and utilize them for PCL, while dynamically updating the prototypes. The intra-class consistency and inter-class separability are maximized by optimizing ICL and PCL.

where $y_i$ is the label of the ground truth, and $\tilde{z}_i = \frac{z_i}{\|z_i\|}$ represents the feature normalization operation. $\tilde{z}_i \cdot \tilde{z}_j$ denotes the inner (dot) product between the $i$-th and $j$-th proposal in the projected hypersphere denotes the inner (dot) product. $\tilde{z}_i \cdot \tilde{c}_j$ denotes the inner product between the $i$-th proposal and the $j$-th class prototype feature. $\tau$ denotes the temperature parameter.

**Sample selection strategy.** Unlike image classification, which uses the semantic information of the whole image, our model classifies based on the classification signals in the candidate boxes generated by RPN. Considering that most of the candidate boxes may be offset from the object instance, the information in the candidate box does not describe the corresponding instance accurately enough, so the contrast samples are screened. We use the Intersection over Union (IoU) score $u_i$ between proposal and its matching ground truth bounding box to select the corresponding samples. Referring to most of the IoU threshold screening methods, we take the samples with $u_i$ greater than 0.7 as the foreground instances of the $N^+$ samples. At the same time, $N^-$ samples with $u_i$ less than 0.3 are selected as background samples. By selecting the samples, we can obtain the instance objects containing more information and reduce the background interference in the comparison. Besides, the selecting eliminates most of the samples and reduces the amount of calculation of the model.

**Decoupling strategy for positive and negative samples.** Inspired by decoupled contrastive learning [10], we further modify the contrastive loss. Decoupled contrastive learning uses a large number of experiments to prove that the currently widely used cross-entropy (InfoNCE) [15] loss has obvious positive and negative coupling effects, which reduces the effect of the model in small batch learning and affects the training efficiency of the model. Therefore, we use the decoupling strategy of positive and negative samples in the contrast loss. By directly removing the similarity between the positive sample pairs in the denominator, the ratio of the sum of similarities for all positive sample pairs to the sum of similarities for all negative sample pairs can be computed directly. This allows the model to optimize positive and negative sample pairs separately, decoupling their influences. And the training efficiency of contrastive learning is improved.

Subsequently, we use the Prototype Calibration Module (PCB) in [13] to further refine our classification scores. And the classifier of cosine similarity in TFA is adopted in the fine-tuning step.

**C. Prototype Updates**

Since the representation of the image is constantly updated during the training process, the prototype feature embeddings need to be maintained to make the obtained prototype-like feature embeddings more representative. Therefore, we dynamically update the prototypes during training. The prototype update part is shown in Figure 2. For $K$ classes, their initial prototypes are first obtained. The backbone network is used to extract original image features for a given support set $S$, and then RoI Pooling and ground-truth box are used to obtain the object instance embedding representation $f_i$. The initial prototype representation of each class is obtained by means of the mean value. The formula is:

$$c'_K = \frac{1}{|S_K|} \sum_{(f_i,y_i) \in S_K} f_i.$$  \hspace{1cm} (3)

Since the candidate boxes may be biased and contain a lot of background noise, we screen the $N$ features used to update. The $B$ RoI features are selected by the IoU score $u_i$ of the prediction box and the ground truth of the corresponding prediction class greater than threshold $\phi$. 

687
Fig. 2. Each subgraph consists of a class prototype feature and its corresponding class RoI features as nodes, and the edge is the cosine similarity between the feature vectors. The information of RoI features is aggregated into the corresponding class prototypes by the way of GCN.

$K$ subgraphs $G = (V, E)$ are constructed by using the class prototypes and the selected $B$ RoI features, where $V$ and $E$ represent the node set and edge set. The node set $V$ of each subgraph consists of a class prototype and the RoI features of its corresponding class. Edges are constructed between prototype and RoI features, and between RoI features to represent the relationships between them. Thereinto, the information from the RoI features of the associated class can be transferred to the prototype to update its features by creating edges between the prototype and RoI features. By constructing edges between RoI features, different RoI feature information can be used to enrich the original RoI feature information. The adjacency matrix $A$ is used to represent this relationship. In general, considering that the same class have more similar features, we adopt cosine similarity to obtain the similarity between prototype and RoI features, and between RoI features, thereby obtaining the adjacency matrix $A$. Then, it is normalized to acquire the normalized adjacency matrix $A'$. The formula is:

$$A^K = \left( \frac{c^T}{\|c\|_2} \right) x^K, \quad (4)$$

$$A' = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}, \quad (5)$$

where $D$ denotes the diagonal degree matrix and $D_{ii} = \sum A_{ij}$. For simplicity, $A^K$ is simply denoted as $A$, and the similarity calculation between RoI features is also calculated by referring to (4).

Using the feature similarity size provided by the adjacency matrix $A$, we adopt GCN to aggregate the feature information of the candidate boxes into the corresponding class prototypes. For each prototype $c_K$, the effect of each layer of GCN is equivalent to the weight sum of its corresponding class’s RoI features, and each prototype is updated as follows:

$$c_K = \frac{1}{B_K} c'_K + \sum_{x^K_i \in x^K} A' x^K_i, \quad (6)$$

D. Loss Function

The loss function consists of the classification and bounding box loss functions of RPN and RCNN in Faster R-CNN, as well as the ICL and PCL of the contrastive learning module.

$$L = L_{rpn} + L_{cls} + L_{reg} + \lambda_{icl} L_{ICL} + \lambda_{pcl} L_{PCL}, \quad (7)$$

$\lambda_{icl}$ and $\lambda_{pcl}$ are set to 0.5 respectively, to balance the loss.

IV. EXPERIMENTS

In this section, we first provide a description of the few-shot object detection datasets and detection settings. Following this, we will present our detection results and ablation studies of our approach on Pascal VOC dataset. Finally, we provide the detection results on the MS COCO benchmark.

A. Experimental Setting

Existing benchmarks. We adopt the dataset settings of previous works [4], [13] to ensure that our method can be fairly compared. As for PASCAL VOC, we use three random split groups with 20 classes each, and each group is randomly classified into 15 base classes and 5 novel classes. Each novel class contains $K=1/2/3/5/10$ annotated samples from the combination of PASCAL VOC’s 2007 and 2012’s trainval. The detection ability of novel classes is assessed using nAP50, which is the IoU threshold with an average precision of 0.5 for the novel classes. For MS-COCO, we split the 80 classes into two separate datasets: a base class dataset consisting of 60 classes, and a novel class dataset consisting of the remaining 20 classes, where $K=10/30$. Similarly, we evaluate the detection performance of novel classes by employing nAP and nAP75 with different IoU thresholds.

Implementation Details. Our model framework uses Faster R-CNN [3], and ResNet-101 [1] is adapted as our backbone. All experiments are trained on 1 RTX3090 GPU, and the batch size is set to 4. The solver using standard SGD with momentum set to 0.09 and weight decay of 5e-5. During the base training phase, we set the learning rate to 0.005, and adjust it to 0.01 in the fine-tuning phase.

B. Experiments on PASCAL VOC

1) Comparisons with State-of-the-art Methods: Our method is compared with the previous state-of-the-art methods on the experimental results of three random splits of PASCAL VOC, as shown in Table I. It can be seen that FS-IPCL demonstrates better performance than existing FSOD methods in the majority of experimental conditions. Specifically, for Novel Split 1, our method improves by 3.8% (40.2% vs. 44.0%) in the 1-shot setting, 3.6% (53.6% vs. 57.2%) in the 2-shot setting, an improvement of 0.8% (58.2% vs. 59.0%) in the 3-shot setting and 3.6% (53.6% vs. 64.3%) in the 1-shot setting, 0.7% (63.6% vs. 64.3%) in the 10-shot scenario, our method is on par with the current best performing method and a 0.8% improvement over the second best performing method (59.7% vs. 66.5%). In the remaining two splits, although the existence of FS-IPCL is lower than other methods, the
overall average performance of our method is the best, with an improvement of 1.4%.

In the setting of 2-shot of split1, we visualize the bounding boxes with confidence greater than 0.7, as shown in Figure 3. The successful and failure detection cases are shown to help us analyze the error types. In the successful cases, our model performs well in detection of some novel classes, particularly under challenging conditions such as complex background and low illumination. In the case of failure, there are more missed and false detections for the novel class of small objects and similar objects. The probably reasons are that the multi-scale feature extractor is not used in our approach and there are not many examples to compare in 2-shot.

2) Ablation Study: Ablation for each module of FS-IPCL. The effectiveness of modules showing in the Table II, we confirm the effectiveness of the ICL and PCL modules separately. From the experimental results, we can see that both ICL and PCL improve the effect of using the model alone. However, from the last three rows of Table II, it can be seen that the combination of ICL and PCL has the best effect. It is probably because the ICL compares objects based on local instance structure, while PCL uses global semantics to construct the object and compare global semantic structures information of the object. These two losses are complementary and are able to combine local contrast and global contrast.

Ablation for decoupling positive and negative sample strategy. To further verify the effectiveness of the decoupled positive and negative sample strategy, we conduct explicit experiments to show its performance. Better performance is achieved by decoupling the positive and negative samples in the contrastive loss, as demonstrated in Table III.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1  2  3  5  10</td>
<td>1  2  3  5  10</td>
<td>1  2  3  5  10</td>
<td></td>
</tr>
<tr>
<td>FR2C-1t [16]</td>
<td>ICCV 2019</td>
<td>9.9 15.6 31.6 38.0 52.0</td>
<td>9.4 13.8 17.4 21.9 39.7</td>
<td>8.1 13.9 19 23.9 44.6</td>
<td>24.3</td>
</tr>
<tr>
<td>FSRW [12]</td>
<td>ICCV 2019</td>
<td>14.2 23.6 29.8 36.5 35.6</td>
<td>12.3 19.6 25.1 31.4 29.8</td>
<td>12.5 21.3 26.8 33.8 31.0</td>
<td>25.6</td>
</tr>
<tr>
<td>TFA w/cos [4]</td>
<td>ICML 2020</td>
<td>25.3 36.4 42.1 47.9 52.8</td>
<td>18.3 27.5 30.9 34.1 39.5</td>
<td>17.9 27.2 34.3 40.8 45.6</td>
<td>34.7</td>
</tr>
<tr>
<td>Viewpoint [5]</td>
<td>ECCV 2020</td>
<td>24.2 33.5 42.2 49.1 57.4</td>
<td>21.6 24.6 31.9 37.0 45.7</td>
<td>21.2 30.0 37.2 43.8 49.6</td>
<td>36.7</td>
</tr>
<tr>
<td>MFSR [8]</td>
<td>ECCV 2020</td>
<td>34.7 42.6 46.1 49.4 56.7</td>
<td>22.6 30.5 31.0 36.7 41.3</td>
<td>27.5 32.5 38.2 44.6 50.0</td>
<td>39.1</td>
</tr>
<tr>
<td>QA-FDet [18]</td>
<td>ICCV 2021</td>
<td>41.0 33.2 35.3 47.5 52.0</td>
<td>23.5 29.4 37.9 35.9 37.1</td>
<td>31.2 29.4 37.6 39.8 41.4</td>
<td>36.9</td>
</tr>
<tr>
<td>FSCE [7]</td>
<td>CVPR 201</td>
<td>32.9 44.0 46.8 52.9 59.7</td>
<td>23.7 30.6 38.4 43.0 48.5</td>
<td>22.6 33.4 39.5 47.3 54.0</td>
<td>41.2</td>
</tr>
<tr>
<td>DeFR2C [13]</td>
<td>ICCV 2021</td>
<td>40.2 53.6 58.2 63.6 66.5</td>
<td>29.5 39.7 43.4 48.1 52.8</td>
<td>35.0 38.3 52.9 57.7 60.8</td>
<td>49.4</td>
</tr>
<tr>
<td>Meta Faster R-CNN [19]</td>
<td>AAI 2022</td>
<td>40.2 30.5 33.3 42.3 46.9</td>
<td>28.8 32.0 39.0 37.7 37.4</td>
<td>34.0 32.5 34.4 42.7 44.3</td>
<td>36.9</td>
</tr>
</tbody>
</table>

Ours | This work | 44.0 57.2 59.0 64.3 66.5 | 31.6 39.2 43.5 47.1 51.1 | 38.4 50.1 52.1 58.1 59.8 | 50.8     |

TABLE III
Ablation experiments for decoupling strategy for positive and negative samples in FS-IPCL.

<table>
<thead>
<tr>
<th>Decouple the positive and negative samples</th>
<th>Novel Split 1</th>
<th>2  3  5  10</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>58.3 62.6 65.2</td>
<td></td>
</tr>
<tr>
<td>×</td>
<td>58.6 63.9 66.4</td>
<td></td>
</tr>
</tbody>
</table>

2) Ablation Study: Ablation for each module of FS-IPCL. The effectiveness of modules showing in the Table II, we confirm the effectiveness of the ICL and PCL modules separately. From the experimental results, we can see that both ICL and PCL improve the effect of using the model alone. However, from the last three rows of Table II, it can be seen that the combination of ICL and PCL has the best effect. It is probably because the ICL compares objects based on local instance structure, while PCL uses global semantics to construct the object and compare global semantic structures information of the object. These two losses are complementary and are able to combine local contrast and global contrast.
C. Experiments on MS-COCO

With more categories than Pascal VOC, the MS-COCO dataset contains more complex scenarios, making the detection of the model on the MS-COCO dataset more challenging. We validate the effectiveness of our model on 10 and 30 shots of MS-COCO. The results of the model’s detection in the novel class are presented in Table V. It can be seen that our method has an accuracy improvement of 0.6%~3.1% compared with most of the current methods.

<table>
<thead>
<tr>
<th>Prototype updates</th>
<th>Threshold</th>
<th>Novel Split 1</th>
<th>Novel Split 10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>✓</td>
<td>nAP</td>
<td>56.4</td>
<td>64.2</td>
</tr>
<tr>
<td></td>
<td>nAP75</td>
<td>57.5</td>
<td>63.9</td>
</tr>
<tr>
<td>✓</td>
<td>nAP</td>
<td>59.0</td>
<td>64.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>10-shot nAP</th>
<th>10-shot nAP75</th>
<th>30-shot nAP</th>
<th>30-shot nAP75</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRCN-ft [16]</td>
<td>5.5</td>
<td>5.5</td>
<td>7.4</td>
<td>7.4</td>
</tr>
<tr>
<td>FSRW [12]</td>
<td>5.6</td>
<td>4.6</td>
<td>9.1</td>
<td>7.6</td>
</tr>
<tr>
<td>TFA w/softmax [4]</td>
<td>9.1</td>
<td>8.8</td>
<td>12.1</td>
<td>12.0</td>
</tr>
<tr>
<td>Viewpoint [17]</td>
<td>10.7</td>
<td>6.5</td>
<td>15.9</td>
<td>15.1</td>
</tr>
<tr>
<td>QA-FewDet [18]</td>
<td>10.2</td>
<td>9.0</td>
<td>11.5</td>
<td>10.3</td>
</tr>
<tr>
<td>FSCE [7]</td>
<td>11.1</td>
<td>9.8</td>
<td>15.3</td>
<td>14.2</td>
</tr>
<tr>
<td>Meta Faster R-CNN [19]</td>
<td>9.7</td>
<td>9.0</td>
<td>11.3</td>
<td>10.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>10-shot nAP</th>
<th>10-shot nAP75</th>
<th>30-shot nAP</th>
<th>30-shot nAP75</th>
</tr>
</thead>
<tbody>
<tr>
<td>ours</td>
<td>12.7</td>
<td>12.9</td>
<td>16.5</td>
<td>16.8</td>
</tr>
</tbody>
</table>

V. CONCLUSION

In this work, we have proposed a novel approach by combining instance-wise contrastive learning and prototype contrastive learning to learn a balanced feature space. The instance-wise and prototype contrastive loss have been utilized to maximize the inter-class distance and minimize the intra-class distance of the novel class, and the discriminative features of the novel class have been obtained to improve the classification performance of the model for the novel class. In order to better utilize positive and negative samples for contrastive learning, a contrastive sample selecting scheme and a decoupling approach of positive and negative samples have been employed to improve the contrastive loss. Additionally, the prototypes have been dynamically updated using GCN to produce more representative prototype embeddings. Comparative experiments with several state-of-the-art methods based on meta-learning and fine-tuning have proved the proposed model always achieving competitive results. Our work takes a supervised contrastive learning approach to advance research in FSOD. In the future, we will further explore how to introduce unsupervised contrastive learning into FSOD to drive the development of this field.

ACKNOWLEDGMENT

The research reported in this article was supported by the Development Project of Ship Situational Intelligent Awareness System under the Grant MC-201920X01, the National Natural Science Foundation of China under the Grant No.61991410.

REFERENCES

SWS-NET: An Image Segmentation Framework For Chronic Wounds Based On Self-Supervised Learning

Jiyun Li∗, Sheng Yang† and Chen Qian‡
School of Computer Science and Technology, Donghua University
Shanghai, China

Email: ∗jyli@dhu.edu.cn, †jensenyoungc@outlook.com, ‡chen.qian@dhu.edu.cn

Abstract—Automatic monitoring and evaluation of chronic wounds usually requires massive labeled data sets for segmentation training. Because of the high cost of time and labor, these data are usually difficult to obtain. In order to improve the segmentation effect of the wound image with a small number of labeled samples for training, this paper proposes an image segmentation framework based on self-supervision, summarize a relatively optimal pre-training task for chronic wound image segmentation, minimizes the redundancy between the symmetric network projection output by learning the feature information of two views generated by the same chronic wound image under different distortion transformation, and finally learns valuable knowledge that is conducive to the downstream wound image segmentation task. In addition, this framework also optimize the network structure and loss of the segmentation model. The experimental results show that after self-supervised learning pre-training with a full amount of unlabeled data, the segmentation framework can achieve significant improvement in precision (up to 8%), recall (up to 5%), and MIOU (up to 9%) by fine-tuning with only a small amount of labeled data. This can provide a clear optimization direction for the application of self-supervised learning to specific image segmentation.

Keywords-Chronic Wound; Image Segmentation; Redundancy; Self-Supervised; Pre-Training

I. INTRODUCTION

Chronic wound is a kind of wound that cannot be healed in a short time through regular treatment. In wound healing, clinicians need to continuously measure and evaluate the wound to monitor the healing process and treatment effect. The development of computer technology has brought convenience to the whole work process. Among them, chronic wound image segmentation is an essential step in computer wound measurement. The current mainstream depth convolution neural network (DCNN) performs well in medical image segmentation [1-2].

However, this model based on supervised learning needs a lot of labeled data to train a model with good results. In chronic wounds, obtaining data sets with large-scale pixel labeling is time-consuming, labor-consuming, expensive, and requires a professional clinical experience. In order to reduce the burden of annotation, many methods other than supervised learning have been proposed to improve the labeling efficiency of medical imaging, including semi-supervised learning [3-4], self-supervised learning [5-8], transfer learning [9-10], in which self-supervised learning has obvious advantages over other methods. There is no need for any labeling data during the pre-training phase. It can learn the useful representation of unlabeled data through pre-training tasks to better solve the problem of scarcity of labeled data. Many articles have proved its effectiveness [11-14] on well-known public data sets (such as ImageNet [15]). However, the related research on specific image segmentation, especially chronic wound images, is still relatively rare. So we plan to introduce a self-supervised learning method, targeted optimization for chronic wounds, to improve the segmentation effect when the data on chronic wounds are scarce.

The main contribution of this research is to propose a network framework for chronic wound image segmentation based on self-supervised learning, which we named SWS-NET. In this framework, many unlabeled data are used in the pre-training phase to get useful knowledge for downstream segmentation tasks, which is learned by comparing the similarity of the same image under different distortions from the unlabeled data. Compared with the mainstream segmentation models (such as Unet), it improves the segmentation precision of chronic wound images in the case of sparsely labeled data. In this process, we focus on the effect of the pre-training model trained by different distortion methods on the image segmentation task of chronic wounds. Finally, we summarize a relatively optimal pre-training task for chronic wound image segmentation.

We subsequently developed a small program to realize the segmentation application of chronic wound images on mobile devices such as mobile phones. Therefore, we also made some lightweight modifications to the convolution module of the segmentation network.

II. RELATED WORK

A. Image Segmentation

Currently, the encoder-decoder structure is one of the most popular end-to-end image segmentation frameworks. The full convolution network (FCN) [1] based on this structure has achieved relatively successful image segmentation results, but it reduces the interpretability of the model, resulting in poor segmentation results for specific types of images (such as medical images). With further research, Ronneberger et al [2], proposed an encoder-decoder structure with high applicability, namely U-Net, which provides a more reliable backbone network for wound image segmentation framework. Deeplab [16] is similar to it, but the network structure will be much more complex, which is not convenient for subsequent improvement.

DOI reference number: 10.18293/SEKE2023-134
Since Unet was proposed, it has been concerned and applied by many scholars, such as Attention Unet [17] proposed by O. Oktay et al., and the initial convolution layer introduced by Narinder Singh Punn et al. [18], which further strengthens feature extraction. However, this leads to time-consuming training process, and when annotation data of chronic wound images are scarce, it is easy to over-fit. After that, Francois Chollet et al. proposed an RCA-IUnet network structure [19], which discusses and uses depthwise separable convolutions, integrates the advantages of attention filter, mixing pool, and initial convolution layer, reduces the complexity of convolution neural network structure, and provides a good solution for the segmentation model of this article.

B. Self-supervised Learning

Given the lack of annotation samples, more and more scholars have focused on the research of self-supervised learning.

The agent tasks of early self-supervised learning are usually geometric transformation prediction, flip, rotation angle prediction [20], and jigsaw puzzle [21]. These methods in the initial phase of self-supervision need a well-defined task to be effective for a specific image, and reasonable constraints [22] are needed to prevent the model from obtaining trivial constant (i.e., collapsed) embeddings.

At present, the well-known contrastive learning methods are MoCo [23], SimCLR [13], BYOL [12], SimSiam [24], and Barlow Twins [25]. These methods adopt specific methods to avoid model collapse, which increase the complexity of the model, and rely heavily on the comparison of negative samples. Barlow Twins proposed a new optimization direction to avoid these problems. Inspired by biology [26], remove redundancy between networks, which brings a lot of inspiration to our framework.

III. METHOD

As shown in Figure 1, we have constructed a self-supervised learning framework to explore and learn the multi-dimensional features of unlabeled data sets by pre-training the encoder of the segmentation model and finally fine tune the pre-trained encoder with a small amount of labeled data downstream. This framework is mainly composed of several parts: First, the distortion module generates two distortion views for all images in the same batch sampled from the data set; the second is the encoder module, which will be used for pre-training and applied to the downstream segmentation task; then there is the projection module, which is responsible for projecting the three-dimensional features output by the encoder into the one-dimensional space, and narrowing the distance between positive examples in this space. Finally, the decoder module is used to fine-tune the downstream segmentation tasks and restore the prediction mask image through upsampling.

A. Self-Supervision Phase

Figure 1. The overall architecture of the pre-training phase

![Figure 1. The overall architecture of the pre-training phase](image1)

Figure 2. Example of pre-task in pre-training phase (the top is the original image)
At this phase, we have set up four image distortion methods in the distortion module: rotation&flip, grayscale, crop, and color jitter. By assigning different weights to the probabilities of these distortion modes, we made targeted optimization on the chronic wound image and finally obtained an optimal distortion strategy. The processed image effect is shown in the Figure 2.

The decoder is modified from the original Unet framework (see Figure 3). After the coder trains two distorted pictures with shared parameters, the training data is projected into a one-dimensional space. The projection network consists of two layers, each containing full connection, ReLU activation, and batch normalization processing. Finally, we use the loss to make the correlation matrix of the features from different perspectives close to the identity. The definition of loss as expressed in Eq. 1 and 2.

$$\mathcal{L}_{BT} = \sum_i (1 - C_{ii})^2 + \lambda \sum_{i \neq j} C_{ij}^2$$ (1)

$$C_{ij} \triangleq \frac{\sum_{b = A, B} x_{b, i}^A x_{b, j}^B}{\sqrt{\sum_{b = A, B} (x_{b, i}^A)^2} \sqrt{\sum_{b = A, B} (x_{b, j}^B)^2}}$$ (2)

where \(\lambda\) is a constant, weighing the importance of the first and second terms of loss, and \(C\) is a cross-correlation matrix calculated along the batch dimension between the outputs of two identical networks. \(b\) represents the batch index of the input sample, and \(i, j\) represents the network output's vector dimension. Ultimately, the value of \(C\) will range from -1 to 1, -1 means completely irrelevant, and 1 means complete correlation.

After we use many unlabeled chronic wound data sets for the pre-training of contrastive learning, the improvement effect is still relatively limited compared to classification tasks. From the related paper [27-29], we can know that this is because the comparison of different distortions of the same image is more focused on extracting the global representation. However, it is constrained to improving segmentation effect, such as pixel-by-pixel prediction tasks. To solve this problem, Xiangyun Zhao et al. [30] proposed a pre-training feature extractor using pixel-by-pixel, label-based contrast loss. Through experiments, we found that the loss of Barlow Twins is also suitable for this pre-training strategy. After application, the segmentation effect has been further improved.

B. Segmentation Training Phase

Our segmentation framework is shown in Figure 3, which can be divided into two parts: the encoder and the decoder. The encoder is divided into four downsamples, and the decoder is the corresponding four upsampling. Then, by splicing the fragments in the upsampling process and the downsampling process, The stitching retains more dimension and location information, allowing the following neural network layer to freely choose between shallow and deep features, which is more advantageous for semantic segmentation tasks.

In the convolution module part, each module is composed of two depth separation convolutions [31-32] and one ordinary convolution. We use deep separation convolution to replace the ordinary convolution in the original Unet framework, which makes the network more lightweight and convenient for our fine-tuning and application on mobile devices.

Figure 3. The architecture of the segmentation model
In the loss function part, we use the loss, which combines the binary cross entropy and the dice coefficient, as shown in Eqs 3, 4, and 5. However, the cross entropy loss will be dominated by the class with more pixels for the wound image that only accounts for a small part of the background area. For smaller objects, it is difficult to learn their characteristics, thus reducing the effectiveness of the network. To alleviate this problem, we introduce the Dice coefficient loss. The Dice coefficient calculates the intersection ratio between the segmented prediction result area and the ground truth area, neglects a large number of background pixels, and solves the problem of imbalance between positive and negative samples.

\[ L_{BCE} = -(1 - y)\log(1 - x) - y\log(x) \]  
\[ L_{Dice} = 1 - \frac{2|X\cap Y|}{|X| + |Y|} \]  
\[ L_{Final} = \lambda L_{BCE} + \mu L_{Dice} \]  

In the above formula, x represents the actual pixel value, and y represents the predicted pixel value. X represents the number of pixels of the real wound segmented, and Y represents the number of pixels of the predicted wound because the intersection ratio will increase with the improvement of the effect. In order to conform to the optimization direction of the loss, a subtraction process is made, and the value is kept between 0 and 1. \( \lambda \) and \( \mu \) represent the weight parameter of the loss function.

**IV. Experiment**

In this article, our experiment can be roughly divided into three stages: dataset processing, segmentation experiment, and ablation experiment: The processing of data sets is an important part of self-supervised learning, including the division of training sets and test sets and the setting of pre-tasks; In the part of segmentation experiment, we tested the segmentation effect under different amounts of data, and prove that our proposed framework can improve the accuracy obviously when the labeled data is scarce. Finally, in the ablation experiment part, we verified the influence of the models trained by different distortion methods on the experimental results.

**A. Dataset processing**

Our dataset is from the chronic wound dataset publicly available on Kaggle, with a total of 1010 chronic wound images and their corresponding segmented mask images. In order to use additional chronic wound data for self-supervised pre-training, we used the multi-category dataset of chronic wound images also publicly available on Kaggle, with a total of 2023 images, classified into diabetes foot ulcer, burns, normal, pressure ulcer, skin tear, surgical wound, trauma, and venous wound. It can balance different types of wound samples in the dataset and avoid the abnormal error of segmentation of a particular wound type caused by the scarcity of specific types of samples. We need to divide the data volume of pre-training data to verify its effect under different data volumes. The specific division is shown in the Table 1.

<table>
<thead>
<tr>
<th>Data volume</th>
<th>Test Dataset</th>
<th>Training Dataset</th>
<th>Validation Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>110</td>
<td>144</td>
<td>18</td>
</tr>
<tr>
<td>20%</td>
<td>110</td>
<td>216</td>
<td>36</td>
</tr>
<tr>
<td>30%</td>
<td>110</td>
<td>289</td>
<td>54</td>
</tr>
<tr>
<td>40%</td>
<td>110</td>
<td>360</td>
<td>72</td>
</tr>
<tr>
<td>50%</td>
<td>110</td>
<td>360</td>
<td>90</td>
</tr>
</tbody>
</table>

**B. Segmentation experiment**

The framework is based on a Python 3.8 environment and developed with the TensorFlow V2.6 library. Before the training starts, all the pictures are adjusted to 256x256, the batch size of each training step is set to 8, use the SGD optimizer with a learning rate of 1e-3 and momentum parameter of 0.9. The weight decay parameter of the full connection layer is set to 4e-3. Finally, we use 10%, 20%, 30%, 40%, and 50% data volumes to verify the effect of contrastive learning, and the results are shown in Table II.

<table>
<thead>
<tr>
<th>Data volume</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>pre-training</td>
<td>has</td>
<td>no</td>
<td>has</td>
<td>no</td>
<td>has</td>
</tr>
<tr>
<td>Precision</td>
<td>0.622</td>
<td>0.581</td>
<td>0.784</td>
<td>0.704</td>
<td>0.867</td>
</tr>
<tr>
<td>Recall</td>
<td>0.403</td>
<td>0.311</td>
<td>0.582</td>
<td>0.543</td>
<td>0.709</td>
</tr>
<tr>
<td>MIoU</td>
<td>0.547</td>
<td>0.455</td>
<td>0.693</td>
<td>0.606</td>
<td>0.783</td>
</tr>
</tbody>
</table>

**We have selected precision, recall, and MIoU as our evaluation indicators, and their calculation formula is shown in Eqs 6, 7, 8:**

\[ \text{Precision} = \frac{TP}{TP + FP} \]  
\[ \text{Recall} = \frac{TP}{TP + FN} \]  
\[ \text{MIoU} = \frac{1}{k} \sum_{i=1}^{k} \frac{TP}{FN + FP + TP} \]
From the experimental results, we can see that compared with the model without self-supervised pre-training, the self-supervised framework we proposed can achieve a certain improvement under different fine-tuning data amounts such as 10%, 20%, 30%, 40%, 50%, and can achieve a more significant improvement when the data volume is low, for example, in the case of 20% data volume, it can achieve more than eight percentage points of improvement.

Figure 4. Precision comparison chart

Figure 5. Recall comparison chart

By observing the Figures 4 and Figure 5, we can find that, on the whole, both the precision and the recall rate show more obvious advantages than the model without pre-training under small data volume, but the trend of its curve is still slightly different. With the increase in data volume, the improved range of precision shows a trend of increasing and decreasing. When the data volume is large enough, the precision may even be lower than the model directly trained by supervised learning. With the increase in the amount of data, the increase in recall shows a trend of decreasing at first and then increasing. There is a negative correlation between recall and accuracy. Since the two evaluation indicators, precision and recall rate, have their respective focus, in order to more fairly evaluate the improvement effect under different data volumes, we draw the improvement effect diagram of MIoU (see Figure 6). Through observation, our model has significantly improved when the data volume is small. Although the gap between the precision will be gradually narrowed after the data volume is increased, this more balanced indicator can prove the effectiveness of the self-supervised pre-training framework.

Figure 6. MIoU comparison chart

C. Ablation experiment for pre-task

Our current pre-task applies four mainstream pre-tasks: rotation&flipping, grayscale image, clipping, and color jitter. At the same time, a comparison between the original image and the mask image is added to improve further the learning effect of the pre-task on the segmented task. In this part of experiment, we gradually remove the superimposed distortion method to evaluate the accuracy impact. Through previous experiments, we can achieve more remarkable improvement in 20% of the labeled data sets, making the ablation experiment intuitive. So the ablation experiment is also carried out with 20% data volume. The specific results are shown in Figure 7.

Through vertical comparison, our improved Unet framework can consistently achieve higher segmentation precision than the original Unet framework, which shows that the primary performance of our improved segmentation model exceeds the original Unet model. At the same time, the gap between the original model and our model is gradually reducing with the gradual reduction of distortion. It can also be proved that some of the fine-tuning we have done are targeted optimization for these distortion tasks.

Figure 7. The influence of different image distortion on the result
Through horizontal comparison, we can see that the most important factor affecting the segmentation precision is the contrastive learning between the original and mask images. Moreover, in the segmentation task, the importance of grayscale is no less than the color jitter, which differs from the result of the previous different distortion methods on the classification task. In the classification task, the impact of color jitter accounts for a large proportion, leading to the model not being robust enough to remove some distortion methods. Therefore, a good pre-task is the key to the effectiveness of the self-supervised learning framework. The single-image distortion method may even have the opposite effect.

V. CONCLUSION AND PROSPECT

We propose a network framework for chronic wound image segmentation based on self-supervised learning, alleviate the problem of the scarcity of chronic wound labeling data and provide a better segmentation accuracy for the segmentation of chronic wound images than the mainstream benchmark. In this process, we also discussed the effect of the pre-training model trained by different distortion methods on the downstream segmentation task. Through experiments, the effectiveness of our experiment has been proved, which provides reference methods and ideas for further research and application.

However, there are also some areas that need to be improved during the experiment. According to the experimental data, the recall rate has been at a relatively low level. What causes the low recall rate and what optimization methods can improve the recall rate. At present, there are two reasons to guess: one is the image quality problem in the data set, and the other is the loss function in the fine-tuning stage, which needs to be further explored by later generations.

REFERENCES

BDC-FR: Faster R-CNN with Balanced Domain Classifier for Cross-Domain Object Detection

Shouhong Wan, Rui Wang, Peiquan Jin, Xuebin Yang
School of Computer Science and Technology
University of Science and Technology of China
Hefei, China
wansh@ustc.edu.cn, wr666@mail.ustc.edu.cn, jpq@ustc.edu.cn, yangxb@mail.ustc.edu.cn

Abstract—Object detectors trained with massive labeled data often suffer performance degradation in some particular scenarios with data distribution gap. Domain classifier is a commonly used method in the existing domain adaptation algorithms to alleviate the domain discrepancy. However, it has problems of instability in training, difficulty in obtaining optimal solutions and converging to an equilibrium point. To tackle this issue, we propose a novel balanced domain classifier (BDC), which not only eliminates domain discrepancy but also makes the domain classifier and the feature extractor maintain equilibrium during the adversarial learning. Furthermore, we propose an appropriate learning rate adjustment strategy, which makes the detection model converge to an equilibrium point more stably and more rapidly. Based on the domain-invariant region proposal network, we propose a cross-domain object detection model called Faster R-CNN with Balanced Domain Classifier (BDC-FR). The experimental results show that BDC-FR can effectively improve the performance of the cross-domain object detection model.

Index Terms—object detection, domain adaptation

I. INTRODUCTION

Applying the object detection models such as Faster R-CNN [1] trained on one image dataset directly to another image dataset will lead to significant performance drop, because the style, resolution, illumination, etc. of images are different. Conventionally, there are two fundamental data sets in cross-domain object detection problem, source domain dataset (with annotation information) and target domain dataset (without annotation information). There is always a distribution change between two domains, and it is crucial to develop approaches that enable better generalization of object detectors.

Recently, various domain adaptation approaches [2]–[6] have been proposed to solve this problem. To address this issue, these approaches attempt to build invariant feature representation by employing domain classifiers in adversarial learning. However, adversarial learning has been known to be unstable to train due to training instability and sensitivity to hyper-parameters. The relationship between the feature extractor and the domain classifier can easily become unbalanced during this process. When the feature extractor can easily deceive the domain classifier after a few iterations of training, the prediction result of the domain classifier is similar to a random value and cannot provide an effective optimization gradient for the feature extractor. On the other hand, if the domain classifier has a strong learning ability, it can accurately predict the domain label of the image every time, which will cause the phenomenon of gradient disappearance.

To overcome such unbalanced relationship between the domain classifier and the feature extractor, we design a novel balanced domain classifier network, which can effectively make the feature extractor and the domain classifier maintain a balanced state during training. Furthermore, considering that learning rate also has an impact on the convergence of the model, we propose a simple yet effective learning rate adjustment strategy to make the update of network weight more reasonable. Finally, we build a cross-domain detection model called Faster R-CNN with Balanced Domain Classifier (BDC-FR). We conduct several experiments to evaluate BDC-FR in multiple datasets, and the results demonstrate the effectiveness of our model.

The contribution of this work can be summarized as follows: (i) we design balanced domain classifiers to solve the unstable training problem in cross-domain object detection with domain classifier; (ii) we propose a learning rate adjustment strategy to make the detection model converge to an equilibrium point more stably and more rapidly; (iii) we propose a novel cross-domain detection model called BDC-FR and conduct extensive experiments to validate the effectiveness of proposed BDC-FR.

II. RELATED WORK

Unsupervised domain adaptation (UDA) aims to transfer the information learned from a large number of labeled samples in the source domain to the target domain to solve the same problem, while the available samples in the target domain are unlabeled. Chen et al. [2] initially build a method based on Faster R-CNN, which minimizes the domain discrepancy by utilizing domain classifier at image- & instance-level. MeGA-CDA [5] employs category-wise domain classifiers to ensure category-aware feature alignment for learning domain-invariant discriminative features. Zhao et al. [6] strengthen both the classification and localization capabilities of the cross-domain detector by developing fine-grained feature alignment in separate task spaces. Domain classifiers have limited classification ability due to the unstable adversarial training process. In this paper, we propose a balanced domain classifier network to solve the imbalance between the feature extractor and the domain classifier.
III. THE PROPOSED MODEL

A. BDC-FR Model

In this subsection, we overview the architecture of Faster R-CNN with Balanced Domain Classifier (BDC-FR). Fig. 1 illustrates the framework of our proposed model. Our model contains three major components, including the basic feature alignment network, the domain-invariant region proposal network, and the balanced domain classifier.

B. Domain-Invariant Region Proposal Network

Region proposal plays an important role in object detectors. To get better region proposals, we use an RPN domain classifier to minimize domain shift between domains. Specifically, we extend RPN by embedding an RPN domain classifier, and train the classifier in an adversarial learning manner by using a Gradient Reversal Layer (GRL) [7]. The optimization objective of the RPN domain classifier is defined by (1):

$$
\mathcal{L}_{rpn} = -\sum_{i,u,v} [D_i \log R_i^{(u,v)} + (1 - D_i) \log(1 - \log R_i^{(u,v)})],
$$

where $D_i$ denotes the domain label of the $i$th image, and $R_i^{(u,v)}$ is the output of the region-level domain classifier at $(u,v)$ of the RPN feature map.

The basic feature alignment network consists of image-level adaptation and instance-level adaptation. The image-level adaptation and the instance-level adaptation aim to reduce the domain discrepancy in image-level and instance-level, they are defined by (2) and (3):

$$
\mathcal{L}_{img} = -\sum_{i,u,v} [D_i \log F_i^{(u,v)} + (1 - D_i) \log(1 - F_i^{(u,v)})],
$$

$$
\mathcal{L}_{ins} = -\sum_{i,j} [D_i \log N_{i,j} + (1 - D_i) \log(1 - \log N_{i,j})],
$$

where $F_i^{(u,v)}$ denotes the output of the image-level domain classifier at $(u,v)$ of the base feature extractor, $N_{i,j}$ represents the output of the instance-level domain classifier at the $j$th instance of the $i$th image.

In order to ensure that all the domain classifiers are consistent, we design the Double-Consistency Regularization (DCR), which includes two kinds of regulation, namely image- & instance-level consistency regularization and region- & instance-level consistency regularization. The loss of DCR is defined as follows:

$$
\mathcal{L}_{dou_{es}} = \sum_{i,j} \frac{1}{|I|} \sum_{u,v} F_i^{(u,v)} - N_{i,j} \|_2 + \sum_{i,j} \frac{1}{|I|} \sum_{u,v} R_i^{(u,v)} - N_{i,j} \|_2,
$$

where $|I|$ denotes the total number of pixels in the $i$th image and $\| \cdot \|_2$ denotes the Euclidean norm.

C. Balanced Domain Classifier Network

Similar to GANs [8], the domain classifier attempts to accurately distinguish the domain labels of images, while the feature extractor deceives the domain classifier by aligning the features of images. The most important problem is to ensure that the extractor and the classifier are on par to each other. Generally, GANs uses alternating iterative training method to keep the generator and the discriminator balanced. Different from GANs, the detection model with domain classifier is end-to-end and can’t use alternating iterative training method to keep balanced.

In addition to training rounds, network parameters are also a way to control network capability. The parameters of domain classifier network, including the number of convolutional layers, the number and size of convolution kernels and the step size, jointly control the receptive field and feature extraction ability of domain classifier. So the learning ability of the domain classifier can be balanced by controlling these parameters. To obtain the optimal parameters effectively, we propose an iterative control variable method (As shown in Algorithm

Fig. 1. Overview of the proposed Faster R-CNN with Balanced Domain Classifier (BDC-FR). By applying the proposed balanced domain classifier in image-level and region-level adaptation, the domain classifier and feature extractor can maintain balance in the training process.
Algorithm 1: Iterative control variable method

Input:
1. Number of convolutional layer parameters \( n \);
2. Convolutional layer parameters \( p_i \) (\( i \) from 1 to \( n \));
3. Threshold \( t \).

Output: Convolutional layer parameters \( p_i \).

1. Initialize \( p_i \), \( i = 0 \) and current accuracy \( acc_{now} = 1 \);
2. do
3. Fix parameter values other than \( p_i \) and optimize \( p_i \);
4. \( acc_{pre} = acc_{now} \);
5. Calculate the current accuracy of the cross-domain detection model \( acc_{now} \);
6. \( i = (i + 1) \mod n \);
7. while \( |acc_{pre} - acc_{now}| > t \);

1) to search the optimal parameters of the domain classifier network. During each search, only one parameter value is controlled as a variable, and the rest are fixed values. Search the optimal value of the variable parameters, and repeat the above steps iteratively until the model reaches a better solution. By using Algorithm 1, we can find the optimal parameters of domain classifier, then build the balanced domain classifier.

D. Learning Rate Adjustment Strategy

The learning rate is mainly used to control the strength of adjusting parameters of the detection model. When the prediction error is large, the model has a large learning space. Its parameters can be adjusted with a large learning rate to speed up the convergence. When the error is small, the model has converged closely to the equilibrium point. At this time, it only needs a small learning rate to fine tune the parameters. Therefore, we design a learning rate adjustment strategy that is suitable for training based on adversarial learning methods. The specific adjustment is defined by (5):

\[
\alpha = \begin{cases} 
\alpha_{up} + \frac{loss \cdot t_{low} - loss \cdot t_{up}}{t_{up} - t_{low}} \alpha_{up} \cdot \alpha_{low} & \text{if } loss \geq t_{up} \\
\alpha_{low} & \text{if } t_{up} > loss > t_{low} \\
\alpha_{low} & \text{if } t_{low} \geq loss
\end{cases}
\]

where \( \alpha \) is the learning rate of the current training round of the model, and \( loss \) is the prediction error of the current iteration. \( \alpha_{up} \) and \( \alpha_{low} \) are the maximum and minimum learning rates, respectively. \( t_{up} \) and \( t_{low} \) are the upper and lower thresholds of prediction loss.

E. Overall Loss

The overall loss function for training our BDC-FR network can be summarized as follows:

\[
L_{all} = L_{det} + \lambda_{domain}(L_{img} + L_{ins} + L_{rpn} + L_{dou-ca}),
\]

where \( L_{det} \) is the detection loss generated by the Faster R-CNN. \( \lambda_{domain} \) is the trade-off to balance the loss of object detection and adaptive module. In our experiments, we set the value of \( \lambda_{domain} \) to 0.1.

IV. EXPERIMENTS

A. Datasets and Settings

To validate the effectiveness of our proposed BDC-FR, we perform our model on popular image data sets: Cityscapes [9], Foggy Cityscapes [10], SIM 10k [11], and KITTI [12]. We design three different scenario experiments: Adverse Weather Adaptation, Synthetic Data Adaptation, and Cross Camera Adaptation. We implement BDC-FR with Pytorch and use the VGG16 network as the backbone of our model. Besides, in our learning rate adjustment strategy, we set the parameters \( \alpha_{up} = 2e-3 \) and \( \alpha_{low} = 2e-5 \), the parameters \( t_{up} \) and \( t_{low} \) are set to 12 and 1, respectively. We report mAP with an IoU threshold of 0.5 for evaluation.

B. Experiment Results

<table>
<thead>
<tr>
<th>Method</th>
<th>person</th>
<th>rider</th>
<th>car</th>
<th>truck</th>
<th>bus</th>
<th>train</th>
<th>cycle</th>
<th>bicycle</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base FR [1]</td>
<td>24.5</td>
<td>32.7</td>
<td>35.4</td>
<td>12.7</td>
<td>26.7</td>
<td>9.2</td>
<td>9.9</td>
<td>30.0</td>
<td>22.6</td>
</tr>
<tr>
<td>DAF [2]</td>
<td>25.0</td>
<td>31.0</td>
<td>40.5</td>
<td>22.1</td>
<td>35.1</td>
<td>20.2</td>
<td>20.0</td>
<td>27.1</td>
<td>27.6</td>
</tr>
<tr>
<td>SWDA [3]</td>
<td>29.9</td>
<td>42.3</td>
<td>43.5</td>
<td>24.5</td>
<td>36.2</td>
<td>32.6</td>
<td>30.0</td>
<td>35.3</td>
<td>34.3</td>
</tr>
<tr>
<td>MDA [13]</td>
<td>33.2</td>
<td>44.2</td>
<td>44.8</td>
<td>28.2</td>
<td>41.8</td>
<td>28.7</td>
<td>30.5</td>
<td>36.5</td>
<td>36.0</td>
</tr>
<tr>
<td>DIR-FR [4]</td>
<td>36.9</td>
<td>45.8</td>
<td>49.4</td>
<td>28.2</td>
<td>44.6</td>
<td>34.9</td>
<td>35.1</td>
<td>38.9</td>
<td>39.2</td>
</tr>
<tr>
<td>HTCN [14]</td>
<td>47.4</td>
<td>37.1</td>
<td>47.9</td>
<td>32.3</td>
<td>33.2</td>
<td>47.5</td>
<td>40.9</td>
<td>31.6</td>
<td>39.8</td>
</tr>
<tr>
<td>UMT [15]</td>
<td>56.5</td>
<td>37.3</td>
<td>48.6</td>
<td>30.4</td>
<td>33.0</td>
<td>46.7</td>
<td>46.8</td>
<td>34.1</td>
<td>41.7</td>
</tr>
<tr>
<td>MeGA-CDA [5]</td>
<td>37.7</td>
<td>49.0</td>
<td>52.4</td>
<td>25.4</td>
<td>49.2</td>
<td>46.9</td>
<td>34.5</td>
<td>39.0</td>
<td>41.8</td>
</tr>
<tr>
<td>TIA [6]</td>
<td>52.1</td>
<td>38.1</td>
<td>49.7</td>
<td>37.7</td>
<td>34.8</td>
<td>46.3</td>
<td>48.6</td>
<td>31.1</td>
<td>42.3</td>
</tr>
<tr>
<td>BDC-FR</td>
<td>38.2</td>
<td>48.4</td>
<td>52.9</td>
<td>29.8</td>
<td>31.0</td>
<td>43.3</td>
<td>37.1</td>
<td>41.9</td>
<td>42.9</td>
</tr>
</tbody>
</table>

TABLE I

RESULTS OF THE ADVERSE WEATHER ADAPTATION EXPERIMENT.

<table>
<thead>
<tr>
<th>Method</th>
<th>car AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base FR [1]</td>
<td>34.2</td>
</tr>
<tr>
<td>DAF [2]</td>
<td>39.0</td>
</tr>
<tr>
<td>SWDA [3]</td>
<td>42.3</td>
</tr>
<tr>
<td>MDA FR [13]</td>
<td>42.0</td>
</tr>
<tr>
<td>HTCN [14]</td>
<td>42.5</td>
</tr>
<tr>
<td>UMT [15]</td>
<td>43.1</td>
</tr>
<tr>
<td>MeGA-CDA [5]</td>
<td>44.8</td>
</tr>
<tr>
<td>BDC-FR</td>
<td>45.3</td>
</tr>
</tbody>
</table>

TABLE II

RESULTS OF SYNTHETIC DATA ADAPTATION EXPERIMENT.

A large amount of labeled synthetic data is easy to obtain by computer graphics technique. In synthetic data adaptation experiment, our source domain dataset is SIM 10k, which is...
rendered by the game Grand Theft Auto (GTA-V). The target domain dataset is Cityscapes, which is an urban scene dataset from the real world. The results are summarized in Table II. Note that our method reduces the training time. During the model training, our BDC-FR model only trained 9 iterative rounds, which is one less than DIR-FR, which shows that the balanced domain classifier network makes the model converge more rapidly. We argue that the performance degradation of BDC-FR in comparison to DIR-FR is mainly caused by the decrease in the difficulty of the detection task.

### TABLE III

**RESULTS OF CROSS CAMERA ADAPTATION EXPERIMENT.**

<table>
<thead>
<tr>
<th>Method</th>
<th>person</th>
<th>rider</th>
<th>car</th>
<th>truck</th>
<th>train</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base FR</td>
<td>43.3</td>
<td>28.6</td>
<td>73.9</td>
<td>13.6</td>
<td>14.0</td>
<td>34.7</td>
</tr>
<tr>
<td>DAF [2]</td>
<td>40.9</td>
<td>16.1</td>
<td>70.3</td>
<td>23.6</td>
<td>21.2</td>
<td>34.4</td>
</tr>
<tr>
<td>MDA FR [13]</td>
<td>53.0</td>
<td>24.5</td>
<td>72.2</td>
<td>28.7</td>
<td>25.3</td>
<td>40.7</td>
</tr>
<tr>
<td>C2F [16]</td>
<td>50.4</td>
<td>29.7</td>
<td>73.6</td>
<td>29.7</td>
<td>21.6</td>
<td>41.0</td>
</tr>
<tr>
<td>DIR-FR</td>
<td>58.5</td>
<td>37.2</td>
<td>75.4</td>
<td>30.6</td>
<td>18.5</td>
<td>44.0</td>
</tr>
<tr>
<td>BDC-FR</td>
<td>54.4</td>
<td>37.5</td>
<td>73.1</td>
<td>39.0</td>
<td>15.0</td>
<td>44.1</td>
</tr>
</tbody>
</table>

Cameras with different parameters can also cause domain discrepancy even in the same scene. In cross camera adaptation experiment, we evaluate our model on Cityscapes and KITTI. We take Cityscapes as the source domain and KITTI as the target domain. As table III shows, our proposed BDC-FR achieves the best score in most categories. It again takes 9 epochs for BDC-FR to converge, which is faster than DIR-FR (10 epochs).

### C. Ablation Experiment

**TABLE IV**

**RESULTS OF ABLATION EXPERIMENT IN ADVERSE WEATHER ADAPTATION.**

<table>
<thead>
<tr>
<th>Method</th>
<th>$L_{sim}$ &amp; $L_{ins}$</th>
<th>$L_{sync}$</th>
<th>BDC</th>
<th>LR</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (W/o all)</td>
<td>✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
<td>36.0</td>
<td>39.2</td>
</tr>
<tr>
<td>Ours</td>
<td>✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
<td>42.6</td>
<td>42.9</td>
</tr>
</tbody>
</table>

We conduct a ablation study of our proposed method on Adverse Weather Adaptation. Table IV shows the results of ablation study. BDC denotes the balance domain classifier, and LR refers to the learning rate adjustment strategy. The mAP of BDC-FR has improved by 3.4 (from 39.2% to 42.6%) with the balance domain classifier. And the learning rate adjustment strategy allows for a shorter training time, as well as improves the detection accuracy to 42.9%.

### V. Conclusion

In this paper, we propose a cross-domain object detector called Faster R-CNN with Balanced Domain Classifier (BDC-FR). Our key contribution is the balanced domain classifier, which can help the cross-domain object detection model converge steadily by making feature extractor and domain classifier achieve better equilibrium state in training. Furthermore, we propose a learning rate adjustment strategy to improve the convergence of the cross-domain object detection model. In order to verify the validity of the BDC-FR model, we conduct extensive experiments on multiple cross-domain scenarios. Extensive experimental results, as well as ablation studies, demonstrate the effectiveness of the proposed model.

### ACKNOWLEDGMENT

This work is supported by Natural Science Foundation of Anhui Province (Grant No. 2208085MF157).

### REFERENCES


MBR-MDA: Multi-person Behavior Recognition Method Based on Multi Descriptors Aggregations

Yang Luo, Rongheng Lin
State Key Laboratory of Networking and Switching Technology
School of Compute Science(National Pilot Software Engineering School)
Beijing University of Posts and Telecommunications
Beijing, China
{luoyang, rhlin}@bupt.edu.cn

Abstract—Multi-person behavior recognition is an important task in intelligent video surveillance. In this paper, we propose a multi-person behavior recognition method based on multi descriptors aggregations (MBR-MDA) for real-time surveillance scenarios. Our method employs multi-object tracking to obtain consecutive frames of each person, and uses a 2D convolutional network with temporal shift module (TSM) for behavior recognition. To address the limitation of 2D convolutional network in capturing global temporal features, we introduce a plug-and-play module called MDA that can be integrated into the 2D convolutional network. By applying data augmentation and embedding the MDA3D module, our method achieves a 4.8% improvement over TSM baseline on the HMDB51 dataset, with only a minimal speed loss of 0.3ms. We evaluate our method on several public datasets and demonstrate that embedding MDA into other methods can also enhance their performance.

Index Terms—intelligent monitoring, behavior recognition, multi descriptors aggregations.

I. INTRODUCTION

Traditional video surveillance methods need regular video supervision, which requires a large workforce and resources. In recent years, intelligent surveillance systems have also flourished with the development of deep learning. Intelligent surveillance systems perform object and behavior recognition using computer vision technology. When abnormal conditions are detected, the system can alert the personnel to handle it.

Despite advancements in deep learning, intelligent surveillance systems still face numerous challenges, including occlusion, illumination variation, pose variation, and background clutter. Multi-person behavior recognition tasks can be particularly challenging, as they involve multiple persons interacting with each other or objects in a scene, making it difficult to accurately and efficiently capture the temporal and spatial features of their behaviors.

In general, there are 3 methods for multi-person behavior recognition. The first method is based on object detection, which identifies the location and behavioral categories of people using methods such as YOLO [1]. This approach has a fast recognition speed, but the lack of temporal information in a single frame. The second method obtains the person location through object detection and extracts key points using a human skeleton model such as STGCN [2]. This method is slow in acquiring human key points, and is very sensitive to the problem of occlusion. The third method obtains each person continuous images by multi-object tracking, and then recognizes human behavior based on frame sequences, which is balanced in speed and accuracy, and can also handle the occlusion situation. Therefore, we choose the third method as the overall technical route.

To address the issue of delayed response time in multi-person behavior recognition systems, this paper proposes using multi-object tracking and behavior recognition based on 2D convolution. To enhance the performance of 2D convolution in behavior recognition, this paper proposes MDA which produces a fusion of features from multi-pooling methods to achieve a better feature representation for image sequence. For time-series feature fusion, a single pooling method compromises the accuracy of information description. This paper employs GemPooling—a generalizable pooling method that aggregates various pooling methods to obtain a more precise description. Based on the MDA method, we propose Multi Descriptors Aggregation by 2D Pooling(MDA2D) and Multi Descriptors Aggregation by 3D Pooling(MDA3D) as methods for enhancing feature representation in this study.

The main contributions of our paper are as follows:

1. The proposed MBR-MDA method employs multi-object tracking and behavior recognition based on 2D convolution to effectively capture temporal and spatial features of multi-person behavior, making it highly suitable for multi-person behavior recognition.

2. The proposed MDA method achieves this by effectively fusing features from multiple pooling methods to provide a better feature representation for image sequences. This not only enhances accuracy but also maintains calculation speed, making it highly effective for real-time behavior recognition applications.

II. RELATED WORK

A. Object Detection

To recognize behaviors in surveillance scenes, it is necessary to locate people through object detection. Object detection are mainly categorized as two-stage method such as Faster-RCNN [3], and one-stage method such as YOLO [1].
One-stage method as YOLO algorithm has been pursuing more optimized speed and relatively practical accuracy in development. Yolov4 [4] proposed the structure of FPN+PAN to achieve a better fusion of the low-level and high-level features. Yolov5 proposed focus operation at the image input and adaptive padding method which achieved better latency and accuracy.

B. Multi-object Tracking

Multi-object tracking associates the trajectories of multiple objects in a video sequence and assigns a unique ID, thus enabling the acquisition of consecutive multi-frame images of a single object. Multi-object tracking first requires the object location through object detection and then matches objects and trajectories based on kinematic features or appearance features.

Bewley et al. proposed the SORT [5] to achieve simple real-time trajectory tracking using the Kalman filter algorithm and the Hungarian algorithm. Wojke et al. introduced the ReID network for extracting the appearance features of objects based on the SORT algorithm. They proposed DeepSort [6] to achieve more accurate tracking through appearance features. Wang et al. proposed the JRE [7] by embedding ReID module into the Yolov3 [8] to achieve multi-object tracking by using only one network. Zhang et al. proposed anchor-free method to realize fair multi-object tracking FairMOT [9], which has better accuracy.

C. Human Behavior Recognition

After obtaining continuous video sequences of each target by object tracking, the video sequences are fed into a behavior recognition network. The following three network structures based on RNN, 3D convolution, and 2D convolution are mainly used for behavior recognition.

Based on RNN for temporal modeling of features, the temporal information is modeled by the recurrent neural network. Shikhar Sharma et al. proposed AttentionLSTM [10] to process continuous video frames with LSTM units and also introduced Attention mechanism in the model to pay more attention to the regions of action changes to improve the accuracy. However the RNN is slow and unsuitable for recognition tasks in surveillance videos.

Based on 3D convolutional network to extract the features of video sequences, the temporal information is modeled by 3D convolution operation. In 2015, Tran D et al. proposed a general network C3D [11] for video recognition, which takes the conventional 2D convolution and pooling operations and puts them into 3D space to effectively obtain the temporal information. Christoph et al. proposed SlowFast [12] which adopted a two-branch network to obtain behavioral features. Slow branch used to model spatial information, and fast branch used to model temporal information.

Based on 2D convolutional networks to extract the overall features of video sequences, such as Wang et al. proposed TSN [13] in 2016, and Zhou et al. proposed TSM [14] based on 2D time series displacement in 2018. TSM captures video motion information by completing the information exchange between frames through temporal displacement.

In contrast to 3D convolution, 2D convolution is weak in modeling temporal information, but it has a fast inference speed. This paper aims to improve the accuracy of 2D convolution while preserving the advantages of 2D convolution calculation.

III. Our Method

A. Behavior Recognition Algorithm Framework

The multi-person behavior recognition algorithm framework based on multi-object tracking and behavior recognition is shown in Fig.1, which mainly includes three parts: object detection, multi-object tracking, and behavior recognition.

Firstly, YOLOv5s is used to obtain the person’s location. The features of different scales are used to calculate the bounding box and category of the object. Finally, the NMS(Non-Maximum Suppression) method filters and gets the object detection results.

The object detection results are fed into the DeepSort multi-object tracking algorithm, which uses Kalman filtering to provide the optimal estimate of the object’s location in the next frame. The ReID network extracts features of the objects, and the Hungarian algorithm is then used to assign object IDs. This allows us to obtain a continuous video sequence of a single object.

Since the size of the bounding box detected by the target changes, the size of the video frame obtained by target tracking is not fixed. To achieve a more stable effect, we use the fixed center and surrounding padding method to align the size. This allows us to obtain a stable video input into the behavior recognition network.

Finally, the continuous frames are input to the behavior recognition network for recognition, and the result is outputted. Fig. 1 shows the behavior recognition effect in an equipment hall, which can recognize the action states of people standing, looking at each other, walking, lying down, holding objects, and more.

We conducted latency statistics for the multi-person recognition framework using 24 frames of 1920x1080 surveillance video input. The three steps of object detection, multi-object tracking, and behavior recognition took 637ms, 835ms, and 1082ms, respectively. Behavior recognition was the most time-consuming step, and recognition time significantly increased when more people appeared in the scene. To improve accuracy and reduce behavior recognition time, we propose an improved 2D convolution-based behavior recognition method in the next section.

![Fig. 1. Multi-person Behavior Recognition Method Based on Multi Descriptors Aggregations.](image)
B. Multi Descriptors Aggregation——MDA

Compared to 3D CNNs, 2D CNNs have the advantages of fewer parameters and faster speed. However, due to information loss in the ability of 2D convolution to acquire temporal features, it is less accurate than 3D convolutional networks for this task. Therefore, we designed a behavior recognition network based on 2D convolution and proposed a multi-pooling feature description aggregation operation based on TSM (Temporal Shift Module). MDA compensates for the loss of 2D convolution in acquiring temporal information, resulting in improved accuracy for behavior recognition.

Fig. 2. Video feature extraction framework TSM-ResNet50.

1) Network Structure: Fig.2 shows the behavior recognition method based on the TSM(Temporal Shift Module). Firstly, adding the temporal shift module to the backbone ResNet50 [15], obtaining the time series’ interactive information, and enhancing the expression of 2D convolution to obtain the T-frame level feature map. In the case of ResNet50 [15] each feature map is usually 2048×7×7 size, and 8 frames are extracted from each video sequence. After obtaining the frame-level feature maps, the feature maps need to be aggregated by dimensionality reduction. Then through the aggregation module, the 1-dim features of each frame will be obtained by the aggregation module, usually directly in the form of AvgPooling2D, then summing and averaging the N frames of the T-frame level feature map. In the case of ResNet50 [15], obtaining the time series’ interactive information method based on TSM(Temporal Shift Module). MDA compensates for the loss of 2D convolution in acquiring temporal information, resulting in improved accuracy for behavior recognition.

2) Multi Descriptors Aggregation by 2D Pooling——MDA2D: Usually, in order to enhance the representation at the video level. AvgPooling2D, then summing and averaging the N frames of the T-frame level feature map, the feature maps need to be aggregated by dimensionality reduction. Then through the aggregation module, the 1-dim features of each frame will be obtained by the aggregation module, usually directly in the form of AvgPooling2D, then summing and averaging the N frames of 1-dim features, and finally obtaining the 1-dim global feature representation at the video level.

Fig. 3. Multi Descriptors Aggregation by 2D Pooling Framework.

pooling, the features change to a one-dimensional tensor f of dimension C. The definition of the pooling operation can be generalized and defined as the following equation.

\[
f = \left[ f_1 ... f_c ... f_C \right]^T, f_c = \left( \frac{1}{|X_c|} \sum_{x \in X_c} x^{p_c} \right)^{\frac{1}{p}} \tag{1}
\]

From the above Eq.1, the average pooling layer \( p_c = 1 \), while the maximum pooling layer \( p_c \rightarrow \infty \). For the Generalized avgPooling layer with generalization, \( p_c \) takes the value of \( (1, \infty) \), expressing the intermediate state between AvgPooling and MaxPooling. Depending on the dataset, different pooling approaches may produce inconsistent differences in effects, e.g., average pooling can obtain information about larger regions in the image. In contrast, Max pooling can focus more on information about focused regions in the image.

As shown in Fig.3, three branches of pooling are adopted for global feature extraction of video sequences, and the output of the features from the pooling layer are downsampled by Dropout, fully connected layer, Batch Normalization, and ReLU in turn. Finally, the three branches’ features are fused by stacking, and the final output is a 1-dim of single-frame feature expression. Assume that each pooling branch outputs features as \( v^b_i, b_i \) proxy for a certain pooling branch, which is selected from MaxPooling, AvgPooling, and GemPooling. And the input frame-level features are \( v_{ori(2d)} \), with dimensions (C, H, W), has the following expression relation defined by Eq.2.

\[
v^{b_i} = Relu(BN(D(W^{b_i}) \cdot v_{ori(2d)})), \\
b_i \in \{\text{Max2D, Avg2D, Gem2D}\}
\tag{2}
\]

\( W^{b_i} \) is the weight parameter of the fully connected layer. \( D(\cdot) \) donate dropout is a common way to mitigate network overfitting in networks, dropping neuron parameters with a certain probability. Finally, the nonlinear expression of the features is boosted by ReLU. As shown in Eq.3 and 4 below, the final 1-dim feature expression of the t frame is obtained by the aggregation operation of each pooled branch feature \( V^{S_t} \), and further obtains the global feature expression by summation averaging \( V_g \) where \( Concat \) denotes the feature stacking operation in the channel dimension.

\[
V^{S_t} = Concat(v^{b_1}, v^{b_2}, \phi^{(b_3)}) \tag{3}
\]

\[
V_g = \frac{1}{T} \sum_{t} V^{S_t} \tag{4}
\]
3) **Multi Descriptors Aggregation by 3D Pooling——MDA3D:** From the single-frame feature aggregation descriptor by 2D Pooling above, we can obtain an aggregated feature for each video sequence frame. And then obtain the global features of the video sequence by final sum & avg operation, which is still not elegant and intuitive enough. The features between T-frames only interact with each other once by final sum & Avg in Eq.4. Therefore, we propose Multi Descriptors Aggregation by 3D Pooling (MDA3D), which is an improvement on MDA2D. The global feature representation of video sequences is further improved by enhancing the feature aggregation capability of the aggregation module through various 3D pooling methods.

As shown in Fig.4, the MDA3D aggregate feature for the input N-frame feature map $v_{ori3d}$ with dimensions (T, C, H, W). Unlike the MDA2D, MDA3D uses 3DPooling to pool the 3D feature map directly, and LeakyReLU is used instead of ReLU. Based on Eq.2, MDA3D method can be obtained by modifying the input, pooling methods, and the activation function as shown in Eq.5 below.

$$v^{b_i} = ReLU(BN(D(W^{b_i}) \cdot v_{ori3d})), \quad b_i \in \{Max3D, Avg3D, Gem3D\} \quad (5)$$

MDA3D’s input dimension is different from MDA2D. Through the 3D pooling operation, which contains four times of information fusion between T-frames while the 2D Pooling contains only once, the temporal information can be aggregated more effectively through the 3D pooling operation. Since the 3D Pooling already aggregates the time dimension, there is no need to sum and average the time dimension by Eq.4. The 1-dim global representation of the video sequence can be obtained by directly performing the stacking operation of Eq.6.

$$V^{S_t} = Concat(v^{b_1}, v^{b_2}, \phi^{(b_3)}) \quad (6)$$

![Fig. 4. Multi Descriptors Aggregation by 3D Pooling framework.](image)

**IV. EXPERIMENTS**

**A. DataSets**

We evaluate improved behavior recognition algorithm on benchmark datasets—HMDB51 [17] and UCF101 [18]. Finally, the algorithm will be applied to a equipment hall monitoring scenario.

The HMDB51 dataset is a video sequence classification data released by Brown University in 2011. HMDB51 contains 6849 segments of samples, divided into 51 categories, and each category contains at least 101 segments of samples. The number of the training set in the experiment is 3570 video sequences, and the number of validation set is 1530 video sequences.

UCF101 is a medium-sized action video dataset collected from YouTube by the University of Central Florida Computer Vision Research Center. UCF101 contains 13,320 video clips with 101 subdivided behavioral action categories. Each of these categories is divided into 25 subsets, containing 4 to 7 groups of actions. The number of the training set is 9537 video sequences, and the number of the validation set is 3783 video sequences.

**B. Training Details**

The behavior recognition network is implemented using the PyTorch 1.8.0. Training and testing on a Linux server with 2 NVIDIA RTX 3090 GPUs.

The video sequences of the training set are averaged into 8 segments, and 1 frame is extracted from each segment to produce 8 frames of video sequences as the network input. Meanwhile, MultiScaleCrop, RandmoFlip, and MixUp are used for data enhancement.

To achieve a better performance in real-world scenarios, we first pre-train the TSM-Resnet50 model on the Kinetics400 dataset, and then load the backbone parameters to finetune on HMDB51 and UCF101 datasets.

For training, batchsize is set to 32 for 30 epochs. The SGD optimizer is used for gradient descent, and the learning rate is updated in a momentum manner. The initial learning rate is set to 0.005 and decremented using the cosine approach. Cross entropy is used as the loss function in classification, and label smoothing is also used to regularize the classifier.

For prediction, we also tested the inference speed of the network by setting batchsize to 1. The average prediction time of model inference was obtained by 300 times inferences after 10 times GPU pre-inferences.

**C. Main Results**

1) **MDA2D Results:** Firstly, the original TSM method was used as the experimental baseline to obtain 0.7051 and 0.9477 Top1-ACC on HMDB51 and UCF101 datasets. After that, the Top1-ACC accuracy of model was improved by adding tricks such as data enhancement, label smoothing, etc., which improved 2.7% and 1.03% on HMDB51 and UCF101, respectively. That indicated these tricks on image classification can also effectively improve video classification model results.

**TABLE I**

<table>
<thead>
<tr>
<th>Models</th>
<th>Params</th>
<th>FLOPs</th>
<th>Latency</th>
<th>Top1-ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSM (baseline)</td>
<td>23.6M</td>
<td>32.9G</td>
<td>8.7ms</td>
<td>0.7051</td>
</tr>
<tr>
<td>TSM+Tricks</td>
<td>31.0M</td>
<td>49.4G</td>
<td>14.8ms</td>
<td>0.7322</td>
</tr>
<tr>
<td>TSM+Tricks+NL</td>
<td>26.7M</td>
<td>32.9G</td>
<td>8.6ms</td>
<td>0.7448</td>
</tr>
</tbody>
</table>

We use the NonLocal attention mechanism module as an experimental control for MDA2D and MDA3D. NonLocal
enhances the feature representation of the TSM model by inlining the attention weights, which can effectively impose spatiotemporal distance constraints in video sequences. Adding the NonLocal module further improves 0.48% and 0.22% on HMDB51 and UCF101, respectively.

The MDA2D proposed in this paper further improves 1.26% on the HMDB51 dataset and 0.21% on the UCF101 dataset. The improvement on HMDB51 exceeds that of NonLocal, and the improvement on UCF101 is almost the same. Meanwhile, MDA2D has fewer parameters and Flops, which also brings faster inference speed. The inference speed by embedding the MDA2D module is basically the same as the baseline but significantly improves accuracy.

2) MDA3D Results: The improved MDA3D module shows in Table III and Table IV below. This module can effectively fuse the global features from multiple pooling feature perspectives and achieves the best Top1-ACC on both HMDB51 and UCF101 datasets, further improving by 2.11% and 0.45%, respectively, exceeding the fusion results with NonLocal. The accuracy of MDA3D outperforms that of MDA2D, which also verifies that the 3D pooling aggregation structure has better global sequence characterization ability.

### Table II

<table>
<thead>
<tr>
<th>Models</th>
<th>Params</th>
<th>FLOPs</th>
<th>Latency</th>
<th>Top1-ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSM(baseline)</td>
<td>23.7M</td>
<td>32.9G</td>
<td>8.7ms</td>
<td>0.9477</td>
</tr>
<tr>
<td>TSM+Tricks</td>
<td>31.1M</td>
<td>49.4G</td>
<td>14.8ms</td>
<td>0.9602</td>
</tr>
<tr>
<td>TSM+Tricks+MDA2D</td>
<td>26.8M</td>
<td>32.9G</td>
<td>8.6ms</td>
<td>0.9601</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>Models</th>
<th>Params</th>
<th>FLOPs</th>
<th>Latency</th>
<th>ACC@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSM(baseline)</td>
<td>23.6M</td>
<td>32.9G</td>
<td>8.7ms</td>
<td>0.7051</td>
</tr>
<tr>
<td>TSM+Tricks</td>
<td>30.9M</td>
<td>49.4G</td>
<td>14.8ms</td>
<td>0.7370</td>
</tr>
<tr>
<td>TSM+Tricks+MDA3D</td>
<td>26.7M</td>
<td>32.9G</td>
<td>9.0ms</td>
<td>0.7533</td>
</tr>
<tr>
<td>TSM+Tricks+MDA3D+NL</td>
<td>34.1M</td>
<td>49.4G</td>
<td>15.1ms</td>
<td>0.7389</td>
</tr>
</tbody>
</table>

4) Feature Space: Using the t-SNE dimension reduction method in the sklearn package, we obtained a feature visualization map of the model. The MDA3D method generates 1536-dimensional features, while the other comparison algorithm models generate 2048-dimensional features. As shown in Fig. 5, we can see that the features generated by the MDA3D method have a better feature representation effect than the benchmark model. The inter-class distance is larger, and the intra-class distance is smaller. Moreover, for some abnormal points like the characteristic effect of running action category 32, the MDA3D method outperforms other methods without mixing categories 42 and 8.

![t-SNE Visual rendering of feature space](image-url)
To demonstrate that the features generated by the MDA3D model have a better representation, we further classified and identified the features using various machine learning classifiers. Table VI shows that all classifiers, including MLP, SVM, KNN, LR, and RF random forest classifiers. On the hmdb51 dataset, the MDA3D feature classification results achieved the best Top1 ACC, with an effective improvement of more than 3% compared to the benchmark. This proves that MDA3D can extract better semantic features.

### Table VI

<table>
<thead>
<tr>
<th>Models</th>
<th>TSM+Tricks</th>
<th>TSM+Tricks+NL</th>
<th>TSM+Tricks+MDA3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>0.729</td>
<td>0.733</td>
<td>0.745</td>
</tr>
<tr>
<td>SVM</td>
<td>0.733</td>
<td>0.716</td>
<td>0.733</td>
</tr>
<tr>
<td>KNN</td>
<td>0.716</td>
<td>0.733</td>
<td>0.744</td>
</tr>
<tr>
<td>LR</td>
<td>0.733</td>
<td>0.744</td>
<td>0.744</td>
</tr>
<tr>
<td>RF</td>
<td>0.748</td>
<td>0.739</td>
<td>0.749</td>
</tr>
</tbody>
</table>

5) **Case Study:** We built a dataset under the electric equipment hall scenario and tested the MDA method on the dataset.

### Table VII

<table>
<thead>
<tr>
<th>Models</th>
<th>ACC@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSM</td>
<td>0.7615</td>
</tr>
<tr>
<td>TSM+Tricks</td>
<td>0.7824</td>
</tr>
<tr>
<td>TSM+Tricks+MDA2D</td>
<td>0.7947</td>
</tr>
<tr>
<td>TSM+Tricks+MDA3D</td>
<td>0.8035</td>
</tr>
</tbody>
</table>

The results are shown in the Table VII above: MDA3D method adopted in our dataset has also significantly improved (+4.2%) compared with the original TSM baseline.

Fig. 6. Multit-person Behavior Recognition based on TSM-MDA3D.

We use a multi-person behavior recognition framework based on MDA3D to recognize behavior. The left picture shows a sitting posture that is occluded, making it difficult to extract the skeleton points of the entire body and accurately recognize the action. The right picture shows standing and walking behaviors that are challenging to distinguish by single-frame detection. The MBR-MDA method effectively recognizes these cases and improves accuracy using MDA while maintaining inference speed.

### V. Summary

In this paper, we propose a multi-person behavior recognition method based on multi descriptors aggregations (MBR-MDA) for real-time surveillance scenarios.

To address the limited ability of 2D convolution to model temporal features in video sequences, we propose Multi Descriptors Aggregation by 2D Pooling and 3D Pooling.

Our method is evaluated on HMDB51 and UCF101 datasets, and we demonstrate that MDA3D can effectively generate semantics features. With a slight increase in parameters (+3.1M) and inference time (+0.3ms), our method improves classification Top1-ACC accuracy by 4.8% compared to the baseline. Compared to the Non-local method, our method has fewer parameters, shorter inference time, and higher accuracy, making it suitable for real-time behavior recognition models.

### References


