DMSVIVA 2024

Proceedings of the 30th International DMS Conference on Visualization and Visual Languages

October 29 to 30, 2024
Larkspur Landing South San Francisco Hotel,
USA and KSIR Virtual Conference Center, USA

Copyright ⓒ 2024 by KSI Research Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

DOI: 10.18293/DMSVIVA2024

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

PROCEEDINGS

DMSVIVA2024

The 30th International DMS Conference on Visualization and Visual Languages

Sponsored by

KSI Research Inc. and Knowledge Systems Institute, USA

Technical Program

October 29 to 30, 2024

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

Organized by

KSI Research Inc. and Knowledge Systems Institute, USA

Copyright ⓒ 2024 by KSI Research Inc. and Knowledge Systems Institute, USA

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

ISBN: 1-891706-60-8 (print)

ISSN: 2326-3261 (print)
2326-3318 (online)

DOI: 10.18293/DMSVIVA2024

Additional copies can be ordered from:
KSI Research Inc.
156 Park Square Lane
Pittsburgh, PA 15238 USA
Tel: +1-412-606-5022
Fax: +1-847-679-3166
Email: dms@ksiresearch.org
Web: http://ksiresearch.org/ seke/dmsviva24.html

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

DMSVIVA2024

The 30th International DMS Conference on Visualization and Visual Languages

October 29 to 30, 2024

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

Conference Organization

DMSVIVA2024 Conference Chair

Walter Balzano, University of Naples, Italy; Conference Chair

DMSVIVA2024 Steering Committee Chair

Shi-Kuo Chang, University of Pittsburgh, USA; Steering Committee Chair

DMSVIVA2024 Steering Committee

Paolo Nesi, University of Florence, Italy; Steering Committee Member
Kia Ng, University of Leeds, UK; Steering Committee Member

DMSVIVA2024 Program Co-Chairs

Martin Erwig, Oregon State University, USA; Program Co-Chair
Maiga Chang, Athabasca University, Canada; Program Co-Chair

DMSVIVA2024 Program Committee

Danilo Avola, University of Rome, Italy
Andrew Blake, University of Brighton, UK
Paolo Bottoni, Universita Sapienza, Italy
Bernardo Breve, University of Salerno, Italy
Loredana Caruccio, University of Salerno, Italy
Maiga Chang, Athabasca University, Canada
WilliamCheng-Chung Chu, Tunghai University, Taiwan
Stefano Cirillo, University of Salerno, Italy

Mauro Coccoli, University of Genova, Italy
Francesco Colace, University of Salerno, Italy
Gennaro Costagliola, University of Salerno, Italy
Mattia DeRosa, University of Salerno, Italy
Vincenzo Deufemia, University of Salerno, Italy
Tiansi Dong, Bonn-Aachen International Center for Information Technology, Germany
Martin Erwig, Oregon State University, USA
Larbi Esmahi, Athabasca University, Canada
Rita Francese, University of Salerno, Italy
Kaori Fujinami, Tokyo University of Agriculture and Technology, Japan
Angela Guercio, Kent State University, USA
Pedro Isaias, University of Queensland, Australia
Jun Kong, North Dokota State University, USA
Robert Laurini, University of Lyon, France
Weibin Liu, Beijing Jiao Tung University, China
Mark Minas, Universität der Bundeswehr München, Germany
Andrea Molinari, University of Trento, Trento, Italy
Eloe Nathan, Northwest Missouri State University, USA
Paolo Nesi, University of Florence, Italy
Max North, Southern Polytechnic State University, USA
Michela Paolucci, University of Florence, Italy
Giovanni Pilato, Italian National Research Council, Italy
Giuseppe Polese, University of Salerno, Italy
Elvinia Riccobene, University of Milano, Italy
Peter Rodgers, University of Kent, UK
Domenico Santaniello, University of Salerno, Italy
Weiwei Xing, Beijing Jiao Tung University, China
Atsuo Yoshitaka, JAIST, Japan
Tomas Zeman, Czech Technical University, Czech Republic
Yang Zou, Hohai University, China

Publicity Co-Chairs

Danilo Avola, University of Rome, Italy; Publicity Co-Chair
Francesco Colace, University of Salerno, Italy; Publicity Co-Chair

FOREWORD

On behalf of the Program Committee of the *30th International DMS Conference on Visualization and Visual Languages (DMSVIVA2023)*, we would like to welcome you. This conference aimed at bringing together experts in visualization, visual languages, distance education and distributed multimedia computing, providing a forum for productive discussions about these topics.

We would like to thank all the authors for their contributions. We also would like to thank all the Program Committee members for their careful and prompt review of submitted papers.

We would like to thank the Steering Committee Chair Professor Shi-Kuo Chang for his guidance and leadership throughout the organization of this conference. The assistance of the staff at KSI Research is also greatly appreciated, which made the review process smooth and timely.

Finally, we would like to thank you all for joining us in DMSVIVA2024, and really appreciate your participation and your desire to support the community year by year.

Martin Erwig, Oregon State University, USA; Program Co-Chair and
Maiga Chang, Athabasca University, Canada; Program Co-Chair

Table of Content

Session I

Session II

Notes: (S) denotes a short paper.

Keynote

AI-supported Smart City, quo vadis?

Robert Laurini
Professor Emeritus
in Information Technologies
University of Lyon
France

Abstract

In today's landscape, the concept of Smart Cities has become mainstream, with numerous urban centers proudly adopting this label. But what exactly does it entail? In this thought-provoking keynote address, I delve into the multifaceted dimensions of Smart Cities, exploring not only various definitions but also the diverse challenges spanning housing, mobility, education, feeding, commerce, health, industry, security, and public participation. At the heart of this transformation, lies the goal of enhancing the quality of life for all citizens. And how do we achieve this? through the seamless integration of information and communication technologies (ICT). From sensor-based Internet of Things (IoT) solutions to cloud management and knowledge extraction and reasoning, these technologies underpin the Smart City vision. Indeed, artificial intelligence emerges as a pivotal player in this approach. Many believe it holds the key to realizing our ultimate aspirations. During this address, I will survey AI potentialities, including knowledge management, case-based reasoning, and deep learning. However, I must also acknowledge the caveats and uncertainties that lie ahead, as the promises and pitfalls of Smart Cities are explored, envisioning a future where technology harmonizes with human well-being and sustainable urban living. The talk will be concluded by the importance of territorial intelligence, combining intelligently artificial intelligence and human collective intelligence for more sustainable cities.

About the Speaker

Born in 1947, Robert Laurini (aka Roberto) holds two doctorates, one in 1973, and the other in 1980, both in information technologies awarded by the Claude Bernard University of Lyon, France. He speaks fluently French, English, Italian and Spanish. Throughout his career, he primarily worked at INSA-Lyon (University of Lyon), eventually achieving the status of distinguished professor. However, in 1976-77, he spent an entire year as a research associate at the Martin Centre of the University of Cambridge in the United Kingdom. In addition, in 1986-87, he served as a visiting professor at the University of Maryland, College Park, USA. Between 1995 and 2005, he held a part-time position at IUAV University in Venice, Italy. Since 2011, he has been retired and holds the title of professor emeritus**.**

Small Text Object Detection with Pyramid Pooling and Multi-Scale Feature Enhancement

Zhu Yuan[1], Jin Yao[1], Chengjun Liu[1], Wuxuan Tang[2]
[1]Guangzhou Metro Design & Research Institute Co., Ltd., Guangzhou, China
[2]Institute of Intelligence Science and Technology, School of Computer Science and Software Engineering,
Hohai University, Nanjing, China
{yuanzhu, yaojin, liuchengjun}@dtsjy.com, tangwx2001@hhu.edu.cn

Abstract

Detecting small text objects has been a key focus in object detection research due to their unique characteristics: small size, limited semantic information, susceptibility to interference in complex scenes, and tendency to be easily obscured, among others. At present, there are still two common issues in representative object detection models: First, small objects are easily interfered by the background or other objects, and second, multi-layer feature networks cause the loss of small object feature information. To address these challenges, this paper proposes an improved version of the DBNet model by introducing two modules: the contextual information fusion module SPP-CIF and the multi-scale feature enhancement module DA-MSFE. SPP-CIF fuses global and contextual information, by replacing the pooling layer of a pyramid with two sequentially concatenated deliated convolutions of small expansion rates, to encode semantic information of high-level features at multiple scales. DA-MSFE employs spatial attention and channel self-attention to select critical features at different scales and locations, and mines and exploits the correlations between channels to enhance and dynamically aggregate multi-scale features. Extensive experiments were conducted on the publicly available datasets MSRA-TD500 and ICDAR2015. The experimental results show that compared to the baseline model, the proposed model exhibits significantly superior performance in terms of the evaluation metrics.

Index terms— Small text object detection, DBnet, contextual information fusion, dual attention

1 Introduction

As an important carrier of information exchange and perception, text exists widely in daily life, such as advertising logos, promotional slogans, traffic signs, etc. Text object is quite unique, as it often located at the edges of images, far away, or in small fonts. Additionally, the factors such as varying aspect ratios, lack of clear closed contours, complex backgrounds, and lighting variations make small text difficult to detect. Consequently, small text object detection has become an important and challenging research topic.

In recent years, researchers have proposed various methods for text detection. Tian et al. [1] introduced a text detection framework with a vertical positioning mechanism called CTPN, which detects text lines within fine-grained text proposals in the convolutional feature map and extracts contextual information, effectively spotting deeply blurred text. Shi et al. [2] designed a directed text detection method, SegLink. It decomposes text into locally detectable segments and links, and enables full convolutional neural networks to perform dense detection at multiple scales through end-to-end training. Zhou et al. [3] proposed the EAST model, which employs multi-scale feature fusion to adaptively handle text of different sizes and predict words or text lines in arbitrary directions and quadrilaterals in complete images. Li et al. [4] introduced PSENet that utilizes a progressive scale expansion network to generate different scale kernels for each text instance, addressing the localization of arbitrarily shaped text. To alleviate the problem of poor detection of curved text, Long et al. [5] proposed a scene text representation, TextSnake, which better handles the detection of curved text. DBNet [6] improved the segmentation effects by using adaptive threshold maps for training and introducing differentiable binarization to solve the gradient non-differentiability problem. More recently, the model Transformer has also been introduced into this field to tackle curved or polygonal scene text detection [7, 8].

However, as one of the representative models for text detection, DBNet still suffer from two signiffcant drawbacks:

DOI reference number: 10.18293/DMSVIVA24-001

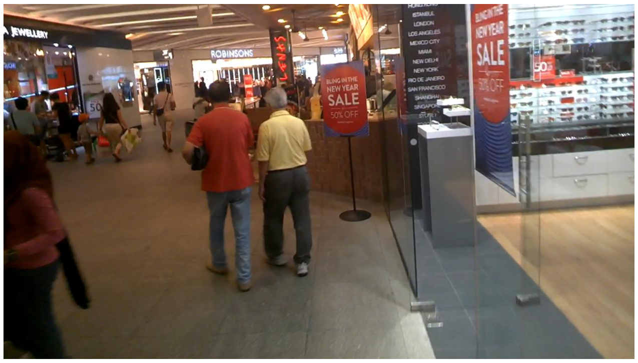


Figure 1: An image in which small texts are disturbed by the background.

Small text can easily be disturbed by the background, as shown in Figure 1. In natural scenes, the background of text images is complex and cluttered, and noise such as lighting interferes with the text detector, reducing the precision of the model and affecting the overall detection precision.

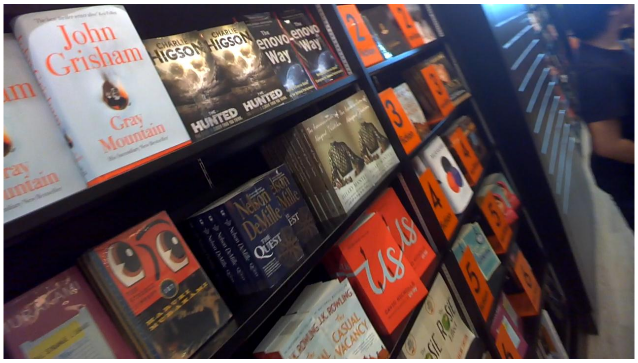


Figure 2: An image with text regions of different shapes.

Text regions with variable shapes are prone to omission. The aspect ratio and size of different text objects in the same image vary greatly, small-sized text is easily missed, and long text is difficult to detect completely. An image with text regions of different shapes is shown in Figure 2.

To address these issues, this paper proposes an improved model on the basis of DBnet, which introduces a Spatial Pyramid Pooling-based Context Information Fusion Module (SPP-CIF) and a Dual Attention-based Multi-Scale Feature Enhancement module (DA-MSFE) to the original model. The main contributions are summarized as follows:

SPP-CIF module: It replaces the pooling layer of the pyramid by applying a tandem dilated convolution in the last layer of the feature extraction network, to semantically encode the high-level features at multiple scales, and fuses the global and contextual information via the global pooling operation, which significantly reduce the interference of background noise.

DA-MSFE Module: Spatial attention weights the fused feature maps and generates feature map weights at different scales, and channel self-attention enhances inter-channel correlation through matrix manipulation and weight calculation. By selecting and aggregating features of different scales and locations, the omission rate of variable text region shapes is considerably reduced.

Experimental Validation: Experimental results on the publicly available datasets MSRA-TD500 and ICDAR2015 demonstrate that the improved model significantly outperforms the baseline model in precision, recall, and F-measure. Particularly, on the MSRA-TD500 dataset, the improved model achieves an increase of 1.4%, 1.6%, and 1.5% in the metrics of precision, recall, and F-measure respectively.

The rest of this paper is organized as follows: Section 2 provides the overall structure of the improved model, Section 3 presents the Spatial Pyramid Pooling-based Context Information Fusion Module, Section 4 introduces the Dual Attention-based Multi-Scale Feature Enhancement Module, Section 5 conducts experiments and result analysis, and finally Section 6 concludes the paper.

2 MODEL ARCHITECTURE

Figure 3 illustrates the architecture of the improved model, which differs from the original DBNet by incorporating two embedded modules SPP-CIF and DA-MSFE. The overall architecture consists of three parts: a feature extraction network, a feature fusion network, and a DBNet detection head. The feature extraction network consists of ResNet50 [9] and SPP-CIF. SPP-CIF is inserted to the last layer of the feature extraction network to fuse local contextual information and global feature information, obtaining the global contextual information of the feature map. The feature fusion network is composed of FPN [10] and DA-MSFE. DA-MSFE is inserted after FPN to enhance the fusion of features from four different scales. The loss function used in this model is consistent with that of DBNet.

3 Spatial Pyramid Pooling-based Context Information Fusion

As the depth of the network increases, the semantic information contained in the feature maps becomes richer. One can effectively capture the contextual information of the image by further extracting semantic information. PSPNet [11] utilized pyramid pooling to fuse features at different scales, thereby reducing the loss of contextual information in sub-regions. PANet [12] employed pyramid structure to extract and fuse contextual information, while also utilizing global pooling to obtain global information.

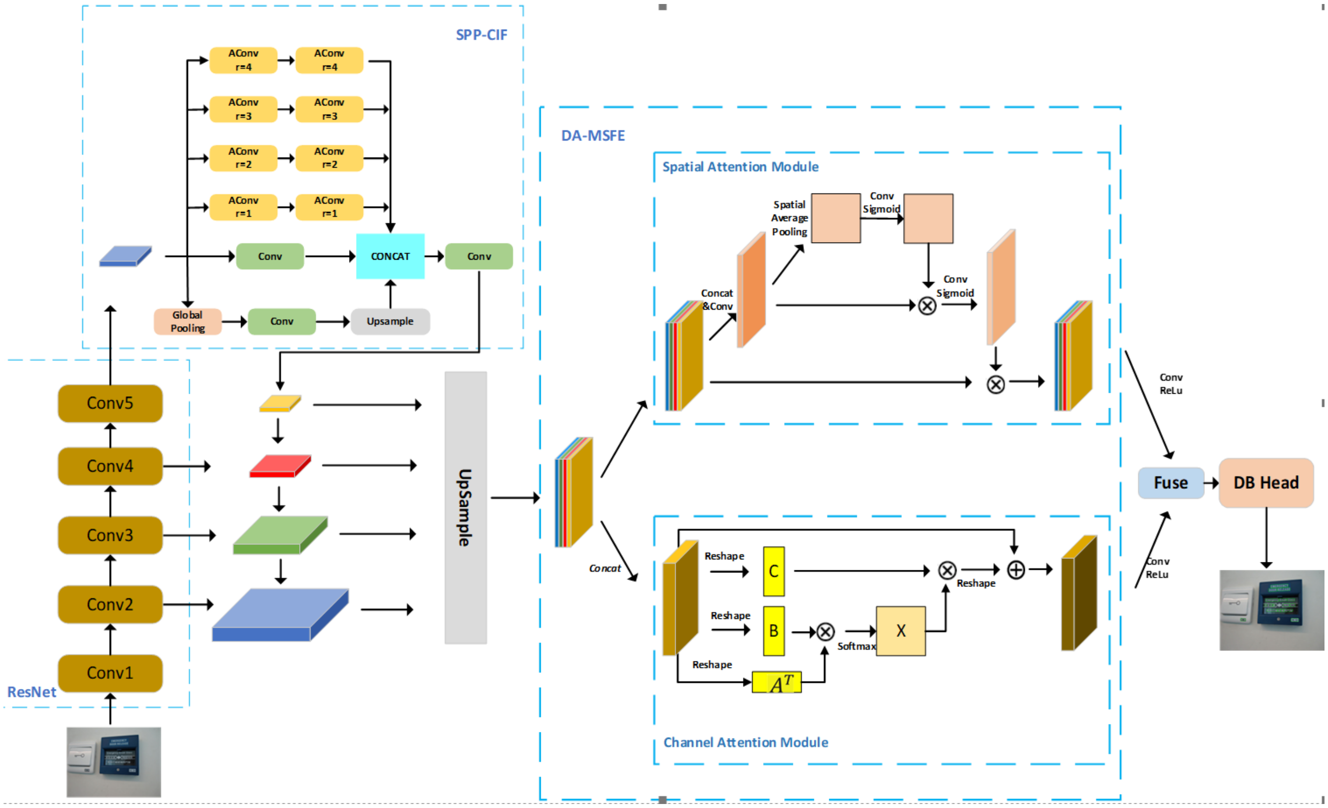


Figure 3: The architecture of the Improved DBNet Model with SPP-CIF and DA-MSFE.

Inspired by these methods, this section proposes a Context Information Fusion Module based on Spatial Pyramid Pooling (SPP-CIF), which can extract information of high-level features from six different receptive fields.

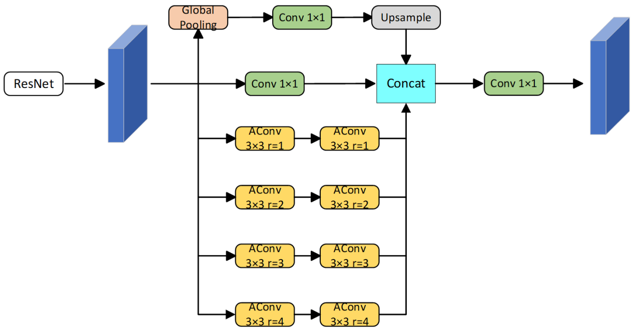


Figure 4: Spatial Pyramid Pooling-based Context Information Fusion Module SPP-CIF.

The network structure of SPP-CIF is shown in Figure 4, and the module consists of a global pooling path, a convolutional path and a dilated convolutional path. The global pooling path is used to acquire global information to further improve the detection performance of the model. The convolutional path retains original feature information. The dilated convolution path employs four dilated convolutions with different dilation rates (r=1, 2, 3, 4) in parallel, increasing receptive fields and obtaining contextual information from diverse regions. Within the path, two dilated convolutions with small dilation rates are sequentially placed to comprehensively extract contextual information from high-level features, particularly focusing on small objects and their surrounding context. The information collected from six different receptive fields are concatenated along the channel dimension and then convolved by one layer to attain the fused feature map.

Specifically, the input feature map F_{in} of SPP-CIF is the output of the last layer of the feature extraction network. F_{in} obtains information with different receptive fields from three respective paths. The process is as follows:

(1) The input feature map F_{in} ($F_{in} \in R^{C\times H\times W}$) is fed to the global pooling path where there is a global average pooling operation, obtaining the global feature descriptor F_{avg} ($F_{avg} \in R^{C\times 1\times 1}$). Then, a convolution operation with kernel size 1×1 is performed to gain the global information F_u ($F_u \in R^{C/2\times 1\times 1}$). Finally, the information is upsampled to achieve F'_u ($F'_u \in R^{C/2\times H\times W}$). This subprocess can be formulated as:

$$F'_u = Up(Conv_{1\times 1}(AvgPool(F_{in}))) \qquad (1)$$

where $AvgPool$ denotes global average pooling, and Up represents upsampling operation.

(2) The input feature map F_{in} is input to the convolution path with kernel size 1×1 to obtain the feature map F' ($F' \in R^{C/2 \times H \times W}$), preserving some of the original information of the feature map. The formula for this subprocess is as follows:

$$F' = Conv_{1\times1}(F_{in}) \tag{2}$$

where $Conv_{1\times1}$ denotes convolution with kernel size 1×1.

(3) The input feature map F_{in} is fed into a parallel dilated convolution pyramid network. This network uses dilated convolutions with kernel size 3×3 and dilation rates r of 1, 2, 3, and 4, respectively. Concatenating dilated convolutions with smaller dilation rates allows for the extraction of context information from different regions and focuses on small objects and their surroundings. This subprocess is formalized as follows:

$$A_i' = AConv_{3\times3,i}(AConv_{3\times3,i}(F_{in})), \quad i = 1, 2, 3, 4 \tag{3}$$

where $AConv_{3\times3,i}$ denotes dilated convolution with kernel size 3×3 and dilation rate of i.

(4) The feature map with six types of information, ${F_u}'$, F', and ${A_i}'$ where $i = 1, 2, 3, 4$, are concatenated along the channel dimension and then fed into a convolution with kernel size 1×1 for further fusion. The output feature map F_{out} (where $F_{out} \in R^{C \times H \times W}$) integrates both global information extracted from high-level feature maps and contextual information. The subprocess is expressed as follows:

$$F_{out} = Conv_{1\times1}(Concat({F_u}', F', A_1', A_2', A_3', A_4')) \tag{4}$$

where $Concat$ denotes concatenation operation along the channel dimension.

4 Dual Attention-based Multi-Scale Feature Enhancement

The feature fusion network outputs feature maps of four different scales, with downsampling factors of 4x, 8x, 16x, and 32x, respectively. These scale feature maps are upsampled to 1/4 of the original image size, and then subjected to feature enhancement by DA-MSFE. These feature maps have varying degrees of importance at different scales, and even within the same scale, the importance varies. Attention mechanisms enable the model to focus more on object regions with valuable information, thereby improving the efficiency and generalization capability. Convolutional Block Attention Module [13] concatenates channel attention and spatial attention to focus on important object regions. Liao et al. [14] construct Adaptive Scale Fusion to learn weights at different scales and spatial locations in the spatial dimension, achieving scale-robust feature fusion. Fu et al. [15] introduce self-attention to assign weights to each pixel in terms of the relationship between input data, thereby capturing dependencies between different positions in the sequence. Inspired by these methods, this section proposes the Dual Attention-based Multi-Scale Feature Enhancement Module (DA-MSFE) to enhance multi-scale features.

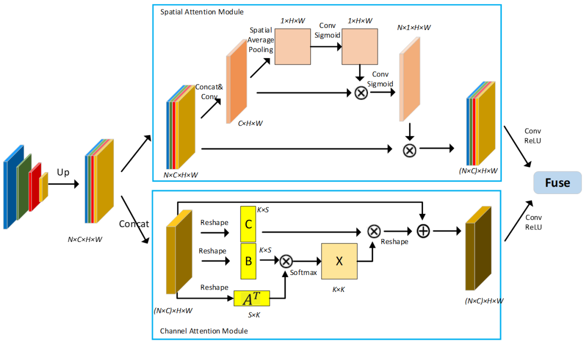


Figure 5: Dual Attention-based Multi-Scale Feature Enhancement Module.

The Dual Attention-based Multi-Scale Feature Enhancement Module (DA-MSFE), as shown in Figure 5, takes as input the feature maps at four different scales output from the feature fusion network. The feature maps at different scales are upsampled to the same scale and fed into two sub-modules. Although upsampling brings these feature maps to the same scale, the features they contain come from different scales. DA-MSFE consists of a spatial attention module and a channel self-attention module. In the spatial attention module, convolutional fusion is performed on the feature maps of different scales. Operations such as average pooling, convolution, and activation are applied to gain the spatial attention weights of the fused feature map. These spatial attention weights are then employed to weight the fused feature map. Furthermore, convolution and activation operations are applied to the enhanced fused feature map to obtain spatial attention weights for the corresponding four scale feature maps. Finally, these weights are utilized to weight the input four feature maps respectively. In the channel self-attention module, by reshaping the feature maps and performing matrix multiplication and weighting operations, each channel is assigned a weight that measures its relevance to other channels. The global information constructed from all channel weights is leveraged to enhance multi-scale features. The two sets of enhanced feature maps are convolved separately, added, and then fused by passing through another convolutional layer, to obtain the enhanced feature map.

The upsampling operation for DA-MSFE is formulated as follows:

$$F_{in}^{i} = UpSample(p_{i+1}), \quad i = 1, 2, 3, 4 \tag{5}$$

where $p_{(i+1)}$(i=1,2,3,4) represents the feature map with a scale of $\frac{1}{2^{i+1}}$ of the original image, $UpSample$ denotes upsampling, i.e., nearest-neighbor interpolation.

The operation of fusing the two enhanced feature maps in DA-MSFE is formalized as follows:

$$F_{out} = Conv_{3\times3,ReLU}\left(Conv_{3\times3,ReLU}(F_{outs}) + Conv_{3\times3,ReLU}(F_{outc})\right) \tag{6}$$

where F_{outs} is the feature map output from the spatial attention module, F_{outc} is the feature map output from the channel self-attention module, and $Conv_{3\times3,ReLU}$ represents the convolution with kernel size 3×3 and $ReLU$ activation.

The following subsections elaborate on the spatial attention and channel self-attention for multi-scale feature enhancement, respectively.

4.1 Spatial Attention for Multi-Scale Feature Enhancement

For the feature maps of different scales upsampled to the same size, the spatial attention module can capture feature information that the model focuses on from various perspectives and receptive fields. For instance, shallow, large-scale features can accommodate more detailed information and small text objects, while deep, small-scale features can capture richer high-level semantic information. To fuse and enhance features from different scales, instead of using simple summation, the spatial attention module DA-MSFE allows the model to autonomously choose important features from different scales and positions, dynamically aggregating features to achieve better integration.

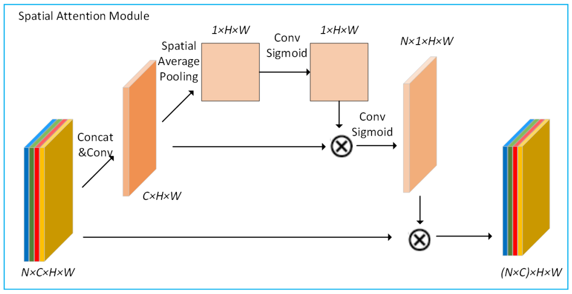


Figure 6: Spatial Attention Module.

The structure of the spatial attention module is shown in Figure 6, and its operation process is described below:

(1) Concatenate the four output feature maps $F_{in}^{i}(F_{in}^{i} \in R^{C\times H\times W}, i = 1,2,3,4)$ to obtain F_{in}, then perform convolution with kernel size 3×3 on F_{in} to obtain the intermediate feature map $F_{in}^{\prime}(F_{in}^{\prime} \in R^{C\times H\times W}, i = 1,2,3,4)$. This subprocess is formulated as follows:

$$F_{in}^{\prime} = Conv_{3\times3}(Concat(F_{in}^{1}, F_{in}^{2}, F_{in}^{3}, F_{in}^{4})) \tag{7}$$

where $Concat$ denotes concatenation operation along the channel dimension.

(2) Perform global pooling on $F_{in}^{\prime}$ to obtain F_{avg} ($F_{avg} \in R^{1\times H\times W}$), then apply a convolution with kernel size 3×3 to F_{avg} followed by a $Sigmoid$ function to obtain the descriptor $M_s(M_s \in R^{1\times H\times W})$. Each position would bear a weight, allowing the model to learn the importance of each position in the fused feature map. This subprocess is formulated as follows:

$$M_s = Sigmoid\left(Conv_{3\times3}\left(AvgPool\left(F_{in}^{\prime}\right)\right)\right) \tag{8}$$

(3) Multiply the spatial descriptor M_s with the feature map $F_{in}^{\prime}$, then apply convolution with kernel size 3×3 to the result followed by a $Sigmoid$ function to obtain the spatial attention A_s for the four scales $A_s(A_s \in R^{N\times1\times H\times W}, N = 4)$. This subprocess is expressed as follows:

$$A_s = Sigmoid\left(Conv_{3\times3}\left(M_s \otimes F_{in}^{\prime}\right)\right) \tag{9}$$

(4) Split the attention weights A_s into four attention weights A_s^{i} corresponding to the four scale feature maps F_{in}^{i}, perform weighting operation for each scale, and then concatenate them along the channel dimension to obtain the weighted feature map $F_{outs} \in R^{(N\times C)\times H\times W}$. This subprocess is formulated as follows:

$$F_{outs} = Concat(A_s^{1} \otimes F_{in}^{1}, A_s^{2} \otimes F_{in}^{2}, A_s^{3} \otimes F_{in}^{3}, A_s^{4} \otimes F_{in}^{4}) \tag{10}$$

4.2 Channel Self-Attention for Multi-Scale Feature Enhancement

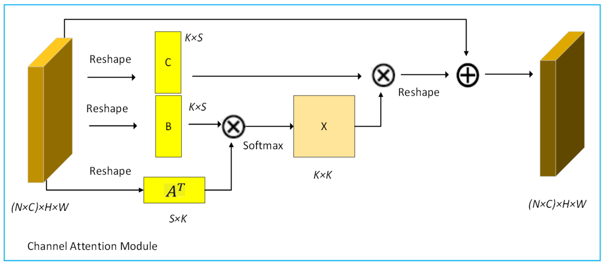


Figure 7: Channel Self-Attention Module.

The spatial attention module only considers the spatial information but neglects the channel information of different scale feature maps. In this section, we introduce the Channel Self-Attention Module to capture the correlation between the channels of these features.The global information consisting of correlations between channels is exploited to enhance the multiscale features.

Unlike traditional channel attention, the Channel Self-Attention Module does not utilize convolution to embed the feature maps. Instead, its implements feature embedding based on self-attention, which enables to fully explore the dependencies between all channels in the feature maps. The structure of the Channel Self-Attention is shown in Figure 7. Its operation process is as described below:

(1) Reshape the concatenated feature map $F_{in}(F_{in} \in R^{(N\times C)\times H\times W})$ into three feature maps A, B, and C, where $\{A, B, C\} \in R^{(K\times S)}$, $(K = N\times C,\ S = H\times W)$.

(2) Transpose feature map A and perform matrix multiplication between feature map B and A^T to obtain a $K \times K$ matrix X'. Apply the $Softmax$ function to X' to obtain the normalized channel attention weight matrix X ($X \in R^{(K\times K)}$). $X_{i,j}$ represents the influence of the i channel on the j channel in the feature map, indicating the weight value. A higher weight value means a higher correlation between the two channels. This subprocess is formulated as follows:

$$X = Softmax(B \times A^T) \tag{11}$$

(3) Perform matrix multiplication between matrix X and feature map C to obtain the weighted feature map A_c. This operation is expressed as follows:

$$A_c = X \times C \tag{12}$$

(4) Reshape the feature map A_c into A_c^T ($A_c^T \in R^{(N\times C)\times H\times W}$), and add it to the input feature map F_{in} to obtain the final output feature map F_{outc}. This subprocess is formulated as follows:

$$F_{outc} = Reshape(A_c) + F \tag{13}$$

5 Experimental Results and Analysis

In this section we designed and conducted ablation experiments and comparative experiments to validate the effectiveness of the proposed model in detection of small text objects. Below, we will introduce the datasets, evaluation metrics, implementation details, and analysis of the experimental results.

5.1 Datasets

Apparently, the types, scales, quantities, and qualities of objects form different datasets can all affect the learning performance of small object detection models. In the experiments, we utilized the following publicly available datasets:

(1) MSRA-TD500: It is published by Huazhong University of Science and Technology in 2012, containing this dataset contains 300 training images and 200 test images, with text boxes labelled as upper-left coordinates, width and height, and deflection angle.

(2) ICDAR2015: It is published by ICDAR in 2015, containing this dataset contains 1000 training images and 500 test images, with text boxes labelled as the four vertices of the polygon.

5.2 Evaluation Metrics

Since the text object is singular, three evaluation metrics are employed to assess the performance of the proposed model: Precision P, Recall R, and F-measure F. The formulas for calculating these metrics are as follows:

$$\begin{aligned} P &= \frac{TP}{TP+FP} \\ R &= \frac{TP}{TP+FN} \\ F &= 2\times\frac{P\times R}{P+R} \end{aligned} \tag{14}$$

where TP refers to the number of positive samples that the model correctly predicts as positive, FP represents the number of negative samples that the model incorrectly predicts as positive, and FN denotes the number of positive samples that the model incorrectly predicts as negative. Precision measures the model's accuracy in predicting positives, indicating the proportion of predicted positive samples that are truly positive. Recall measures the model's coverage of positives, the proportion of positive samples that are successfully predicted by the model. F is the harmonic mean of precision and recall, used to balance precision and recall.

5.3 Experiment Setup and Implementation

We have set up two sets of experiments:

Ablation Experiments: This set of experiments aims to validate the performance of the two modules SPP-CIF and DA-MSFE in detecting small text objects. We use DBNet as the base model and train the following models: **Model 1**: the base model; **Model 2**: the base model with embedded SPP-CIF module; **Model 3**: the base model with the spatial attention part of DA-MSFE embedded; **Model 4**: the base model with the complete DA-MSFE module embedded; **Model 5**: the proposed model, DBNet-SD, i.e., the base model with embedded SPP-CIF and DA-MSFE modules.

Comparative Experiments: We compare the proposed model DBNet-SD with other commonly used text detection models on the ICDAR2015 and MSRA-TD500 datasets to validate its detection performance.

Due to the high randomness of the model initialization parameters, to accelerate model convergence, the backbone feature extraction network pretrained on the SynthText dataset is loaded, and then trained and tested on the datasets MSRA-TD500 and ICDAR2015. Since the dataset MSRA-TD500 has relatively few training images, 400 annotated images from HUST-TR400 are added to it to create a new training set, allowing the model to learn more features and achieve better detection performance.

During training, the input image size is set to 640×640, the number of training epochs is set to 1000, the batch size is set to 32, and the optimizer used is SGD with an initial learning rate of 0.001, following the Poly learning rate schedule. In addition to using random rotation, cropping, and flipping for image augmentation, we also conduct image scaling, skewing, and blurring for preprocessing, enhancing data diversity and further improving the generalization capability of the detection model.

The experiments were conducted on a platform featuring an Intel Gold 6146C CPU and an NVIDIA GeForce RTX 3090 GPU, and the operating system running on the platform is Linux and the CUDA version is 11.0.

5.4 Experimental Results and Analysis

(1)Ablation Experimentals

Table 1. Results of ablation experiments on the dataset MSRA-TD500.

Number	Model	P (%)	R (%)	F (%)
1	**DBNet**	86.1	77.0	81.3
2	**DBNet + SPP-CIF**	87.2	78.0	82.3
3	**DBNet + DA-MSFE_SAM**	86.3	78.0	81.9
4	**DBNet + DA-MSFE**	86.9	78.3	82.4
5	**DBNet + SPP-CIF + DA-MSFE**	**87.5**	**78.6**	**82.8**

The ablation experimental results on the MSRA-TD500 dataset are shown in Table 1. The precision, recall, and F-measure of DBNet are 86.1%, 77.0%, and 81.3%, respectively.

The precision, recall, and F-measure of Model 2 are 87.2%, 78.0%, and 82.3%, respectively. Compared to Model 1, the values of the three evaluation metrics increase by 1.1%, 1.0%, and 1.0%, respectively. This indicates that the introduction of SPP-CIF can integrate contextual and global information and, suppress image background noise, thus enhancing text detection performance and reducing false positives.

The precision, recall, and F-measure of Model 3 are 86.3%, 78.0%, and 81.9%, respectively. Compared to Model 1, the values of the three evaluation metrics increase by 0.2%, 1.0%, and 0.6%, respectively, indicating that the spatial attention module can promote the model's focus on four different scale feature maps, achieving multi-scale feature enhancement.

The precision, recall, and F-measure of Model 4 are 86.9%, 78.3%, and 82.4%, respectively. Compared to Model 1, the values of three evaluation metrics improve by 0.8%, 1.3%, and 1.1%, with a comparatively noticeable improvement in recall. This indicates that the DA-MSFE module can enhance and integrate important information from multi-scale feature maps, and accurately locate text objects in feature maps of different scales, thus alleviating the issue of missed detections and improving detection performance.

The precision, recall, and F-measure of Model 5 are 87.5%, 78.6%, and 82.8%, respectively. Compared to Model 1, the values of three evaluation metrics improve by 1.%, 1.6%, and 1.5%, respectively. Moreover, Model 5 is also superior to Model 2 and Model 4 respectively. This indicates that the embedded modules SPP-CIF and DA-MSFE can work cooperatively in the base model to effectively alleviate the issues of small text object susceptible to background interference and feature loss.

(2) Comparative Experiments The baseline models involved in the comparative experiments include RRD [16] , TextBPN++ [17], PCR [18], FSG [19], EAST, SegLink, and DBNet. To validate the generalization performance of the proposed model DBNet-SD, training and testing of the involved models were conducted on the ICDAR2015 and MSRA-TD500 datasets, respectively.

Table 2. The experimental results on the dataset MSRA-TD500.

Number	Model	Backbone	P (%)	R (%)	F (%)
1	RRD	VGG-16	87.1	73.0	79.4
2	TextBPN++	ResNet-50	86.7	80.8	83.6
3	PCR	ResNet-50	86.5	77.1	81.5
4	FSG	ResNet-50	86.9	**81.0**	**83.8**
5	DBNet	ResNet-50	86.1	77.0	81.3
6	DBNet-SD	ResNet-50	**87.5**	78.6	82.8

The experimental results on the MSRA-TD500 dataset are shown in Table 2. The scores of precision, recall, and F-measure of DBNet-SD are 87.5%, 78.6%, and 82.8%, respectively. Compared to RRD, DBNet-SD shows increases in precision, recall, and F-measure by 0.4%, 5.6%, and 3.4%, respectively. Compared to TextBPN++, the values of precision increases by 0.8%. Compared to PCR, DBNet-SD manifests improvements in precision, recall, and F-measure by 1.0%, 1.5%, and 1.3%, respectively. Compared to FSG, the value of precision improves by 0.6%. The experimental results on the MSRA-TD500 dataset indicate that DBNet-SD has achieved competitive detection performance among the involved baseline models, especially in the metric of precision.

Apparently, the recall and F-measure of DBNet-SD are a bit lower than that of TextBPN++ and FSG on the dataset MSRA-TD500, respectively. This can be attributed to its focus on precision with a stricter detection criterion, leading to potentially miss some true text instances. In contrast, TextBPN++ and FSG might utilize a comparatively lower detection threshold, allowing them to discover a broader range of text instances.

Table 3. The experimental results on the dataset ICDAR2015.

Number	Model	Backbone	P (%)	R (%)	F (%)
1	RRD	VGG-16	85.5	78.9	82.1
2	EAST	PVANet	81.1	72.9	76.8
3	SegLink	VGG-16	72.8	77.0	74.8
4	FSG	ResNet-50	87.8	**83.3**	**85.5**
5	DBNet	ResNet-50	88.0	82.1	84.9
6	DBNet-SD	ResNet-50	**88.6**	82.5	85.4

The experimental results on the ICDAR-2015 dataset are shown in Table 3. The scores of precision, recall, and F-measure of DBNet-SD are 88.6%, 82.5%, and 85.4%, respectively. Compared to RRD, DBNet-SD shows increases in precision, recall, and F-measure by 3.1%, 3.4%, and 3.3%, respectively. Compared to EAST, DBNet-SD manifests increases in precision, recall, and F-measure by 7.5%, 9.6%, and 8.6%, respectively. Compared to SegLink, DBNet-SD demonstrates improvements in precision, recall, and F-measure by 15.8%, 5.5%, and 10.6%, respectively. Compared to FSG, DBNet-SD improves by 0.8% in precision. The results suggest that the improved model DBNet-SD can achieve superior small text object detection performance among the baseline models in scenarios with complex backgrounds.

Similarly, the recall and F-measure of DBNet-SD are slightly lower than that of FSG, which can also be attributed to its focus on the metric of precision.

6 Conclusion

This paper has proposed an improved model DBNet-SD for small text object detection, which integrates two novel modules: SPP-CIF and DA-MSFE into the base model DBNet. SPP-CIF merges the global information and context information from high-level features to promote the capability of understanding the context of objects. DA-MSFE enhances the important regions of feature maps at four different scales, by using spatial attention to dynamically aggregate features and channel self-attention to fully explore the dependencies among all channels in the multi-scale feature maps. Extensive experiments were conducted on publicly available datasets MSRA-TD500 and ICDAR2015. The experimental results show that the improved model significantly outperforms the base model, thus alleviating the issues of background interference, missed detection, and inaccuracy in small text object detection.

In future work, we shall continue to optimize the model DBNet-SD by fine-tuning the network parameters while pursuing the balance between the two evaluation metrics precision and recall.

References

[1] Z. Tian, W. Huang, T. He, et al. Detecting text in natural image with connectionist text proposal network. In *Computer Vision–ECCV 2016: 14th European Conference*, pages 56–72. Springer, 2016.

[2] B. Shi, X. Bai, and S. Belongie. Detecting oriented text in natural images by linking segments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2550–2558. IEEE, 2017.

[3] X. Zhou, C. Yao, H. Wen, et al. East: An efficient and accurate scene text detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5551–5560. IEEE, 2017.

[4] X. Li, W. Wang, W. Hou, et al. Shape robust text detection with progressive scale expansion network. *arXiv preprint arXiv:1806.02559*, 2018.

[5] S. Long, J. Ruan, W. Zhang, et al. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36. Springer, 2018.

[6] M. Liao, Z. Wan, C. Yao, et al. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11474–11481. AAAI, 2020.

[7] X. Zhang, Y. Su, S. Tripathi, et al. Text spotting transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9519–9528. IEEE, 2022.

[8] M. Ye, J. Zhang, S. Zhao, et al. Dptext-detr: Towards better scene text detection with dynamic points in transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 3241–3249. AAAI, 2023.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016.

[10] T.-Y. Lin, P. Dollár, R. Girshick, et al. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125. IEEE, 2017.

[11] H. Zhao, J. Shi, X. Qi, et al. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890. IEEE, 2017.

[12] H. Li, P. Xiong, J. An, et al. Pyramid attention network for semantic segmentation. *arXiv preprint arXiv:1805.10180*, 2018.

[13] S. Woo, J. Park, J.-Y. Lee, et al. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19. Springer, 2018.

[14] M. Liao, Z. Zou, Z. Wan, et al. Real-time scene text detection with differentiable binarization and adaptive scale fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):919–931, 2022.

[15] J. Fu, J. Liu, H. Tian, et al. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154. IEEE, 2019.

[16] M. Liao, Z. Zhu, B. Shi, et al. Rotation-sensitive regression for oriented scene text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5909–5918. IEEE, 2018.

[17] S. Zhang, C. Yang, X. Zhu, et al. Arbitrary shape text detection via boundary transformer. *IEEE Transactions on Multimedia*, 2023.

[18] P. Dai, S. Zhang, H. Zhang, et al. Progressive contour regression for arbitrary-shape scene text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7393–7402. IEEE, 2021.

[19] J. Tang, W. Zhang, H. Liu, et al. Few could be better than all: Feature sampling and grouping for scene text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4563–4572. IEEE, 2022.

Multi-factor CAPTCHA: a usability study

Attilio Albanese, Gennaro Costagliola, Mattia De Rosa, Vittorio Fuccella,
Alfonso Piscitelli, Gaetano Ruggiero, Marco Salierno
Department of Informatics, University of Salerno
Via Giovanni Paolo II, 84084 Fisciano (SA), Italy
{gencos, matderosa, vfuccella, apiscitelli}@unisa.it

Abstract

CAPTCHAs are an important tool for distinguishing human users from computer programs. This paper presents a user study that examined the accuracy, error rate, and user preference for different CAPTCHA schemes, both in a traditional single-factor mode and in a multi-factor mode that combines two different CAPTCHA types in a single challenge. The study involved participants interacting with various CAPTCHA challenges, while collecting data on errors, completion time, and user ratings.

Analysis of the collected data revealed that text-based CAPTCHAs had higher error rates than the other types in both single-factor and multi-factor modes. The checkbox CAPTCHA was found to have the highest accuracy and likability in both authentication modes. Conversely, the audio-based CAPTCHA was the least preferred method by users and also the slowest to complete.

Keywords: *CAPTCHA, Scheme, Single-factor, Multi-factor, Text, Audio, Puzzle, Checkbox, User study*

1. Introduction

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) systems originated as a mechanism to prevent automated access to online services. These systems were developed to distinguish between human users and automated programs, ensuring the security and integrity of online platforms.

Since their introduction, multiple CAPTCHA schemes have been developed, ranging from single-factor to multi-factor approaches. These schemes aim to provide resilience against automated attacks, offering enhanced security measures to protect against unauthorized access.

Single-factor CAPTCHA systems present users with a single verification challenge, such as identifying distorted text, selecting specific images, or ticking a checkbox. While these solutions are simpler and more expedient for the user, they may be less secure, particularly if the challenge is not sufficiently complex or has been compromised by advanced automated programs.

In contrast, multi-factor CAPTCHA systems require users to complete multiple challenges in sequence or in combination. This layered approach significantly enhances security by making it more arduous for automated programs to accurately solve all presented challenges. However, multi-factor CAPTCHA systems can be more time-consuming and may result in increased user friction, as they necessitate additional effort and time to complete the necessary tasks.

This paper presents the findings of a user study that investigated the accuracy, error rate, and user preferences associated with various CAPTCHA schemes implemented in both single-factor and multi-factor authentication modes. A group of voluntary participants were recruited to interact with a custom-built web application that displayed a range of CAPTCHA challenges, which the users were required to accurately solve. Concurrently, the application collected valuable performance data, including the number of errors, completion times, and detailed ratings provided by the participants. The researchers then conducted an in-depth analysis of the gathered data to draw meaningful insights and draw well-supported conclusions about the effectiveness and user experience of the tested CAPTCHA approaches.

The paper is structured as follows: Section 2 provides a review of the relevant literature on CAPTCHA systems. Section 3 outlines the methodology used to design and conduct our user study, detailing the participants, apparatus, procedure, and experimental design. In Sections 4 and 5, the researchers present the analysis of the results obtained during the user study. Finally, Section 6 concludes the paper with the key findings and implications of the study.

DOI reference number: 10.18293/DMSVIVA2024-002

2. Related Work

In 1997, the first single-factor CAPTCHA was introduced, which was based on the input of a text without any form of distortion. With the advent of OCR (Optical Character Recognition), computers could easily solve them, and so, text-based schemes were improved to use distorted text. In [9], researchers conducted a study regarding new trends of CAPTCHAs tracing their development up to 2021, to recognise mechanisms and how to develop new strong and secure schemes. In this context, they described the transition from simpler text-based CAPTCHA to semantic ones with distorted image backgrounds.

The studies presented in the papers *CAPTCHA in Web Security and Deep-Captcha Configuration based on Machine learning* [14] and *A Security Analysis of Text-based Captcha Schemes* [1] highlight the vulnerability of text-based CAPTCHAs. According to the authors, each CAPTCHA should be *Secure and robust, usable, practical, scalable, learnable, universal, localizable, accessible, customizable, and integrated with websites* [1]. Different criteria for a good CAPTCHA give the text *Usability of CAPTCHAs or usability issues in CAPTCHA design* [14]. Some of the results of the text are, that text-based CAPTCHAs can be more difficult for foreigners, the use of colour in a CAPTCHA can have an impact on its usability, security, or both, and whether the length of strings used in a scheme is predictable or not can have interesting implications for both its security and usability. Generally is it notable, that in CAPTCHA there is always a trade-off in usability and security. This can be justified by the fact that with increasing difficulty the CAPTCHAs are harder to solve for humans and computers alike. For each new method of CAPTCHA, the provider has to research an ideal level of difficulty to maximize security, while staying usable.

Still on this subject, researchers of [8] and [11], reviewed different types of CAPTCHA and their alternatives to draw a portrait of the current situation for future studies, being a benchmark in order to direct the upcoming research activities towards inventing more efficient and applicable instances.

A major classification of CAPTCHA schemes was given, for example, text-based such as GIMPY or EZ-GIMPY; image-based such as KeyCAPTCHA, Capy CAPTCHA and ReCAPTCHA; audio-based such as audio ReCAPTCHA or ebay Audio Captcha; etc.

This paper shows both of the drawbacks of CAPTCHAs, like being too complex and annoying for genuine users and accessibility issues, suggesting some alternatives such as audio-based ones for visual impairment.

Usually, CAPTCHAs aim to discern between human and computing systems, due to a challenge. The focus of the researchers was to analyze schemes from an attacker perspective, where the goal is to break them, i.e., to solve the proposed challenge with an automated system and still be recognized as a human. To this end, for each class of CAPTCHA, an attack was described.

Between the analysed schemes, the semantic-based ones are those that seem to tolerate attacks the most as they are far beyond the abilities of machines when it comes to answering only human questions. Moreover, it shows a critical need to design mobile-only CAPTCHAs to preserve the security of growing popular mobile applications, highlighting that mobile users prefer those which involve touch input. Another important drawback found is that users sometimes have either to refresh the CAPTCHA due to the excessive difficulty found in solving it or completely avoid the service. As a last thing, this paper states that other alternatives to CAPTCHA might be more successful.

Among semantic schemes there is SEMAGE [15], which is based on the users' ability to establish semantic relationships between the different images that are proposed to them, and which makes it harder for bots to automatically solve the scheme. As with the other image-based schemes, it is composed of two components: a database of images and a "concept" (the relationship among images). Researchers measured that this scheme was almost on par with Google's reCAPTCHA and much better than the other scheme called Asirra. To demonstrate these results, they conducted a user study with 174 participants. The subject pool was diverse, with most users of a non-computer science discipline. They collected the time taken by each user to complete a challenge for each system (reCAPTCHA, Asirra and SEMAGE schemes, 5 challenges each), and also collected the numbers of successful and failed attempts. The usability of their scheme was evaluated according to the following metrics: time, accuracy and ease of use.

Due to a considerable increase in the ability of bots to solve CAPTCHAs, from 5% to 77%, Microsoft implemented a multi-factor scheme in 2015. However, research showed that even this scheme was weak and could be bypassed with a 44.6% success rate with an average speed of 9.05 seconds. In an attempt to improve it, in [12] an alternative called Smart Captcha has been proposed, which is based on text associated with an image that can be resolved via just selecting the correct checkbox. It has been tested with more than 1000 people of which 700 gave positive feedback (with a score between 90 and 100) with several users that solved it in less than 5 seconds.

In [13] an upgraded version of [12] called Smart Captcha Version 1 is presented, which uses text (with alphanumeric characters) and a puzzle composed of a fragmented and dislocated image. The scheme was offered to more than 1000 people of which around 400 gave positive feedback (with a score between 90 and 100) and most of them solved it in less than 10 seconds. The scheme was found to be clear, readable and easily solvable for humans.

In [2] a new type of scheme is introduced which combines text and images. The goal is to efficiently use image recognition with a challenge that is easy for humans but hard for automated programs. This uses a database filled with thousands of images (collected from popular search engines) and text labels. When the user has to access the service, an image is fetched along all with four text labels for verification, labels scattered over the underlying image. The user, within three attempts, has to recognize the image and select the correct label to prove he is human.

Another multi-factor scheme is proposed in [6]. The scheme is composed of two layers: the first one is static and includes a face recognition CAPTCHA; the second one is randomly chosen from a text-based, video-based or audio-based CAPTCHA.

3. User study

We conducted a user study to compare the time required and accuracy in completing multi-factor CAPTCHA challenges across different modes. Additionally, we included a comparison between single-factor and multi-factor CAPTCHA schemes. In the experiment, participants were asked to solve twelve pairs of CAPTCHA challenges. ~~Table~~ 1 ~~shows the order of modes for each participant.~~ [si fa riferimento alla tabella nella pagina seguente]

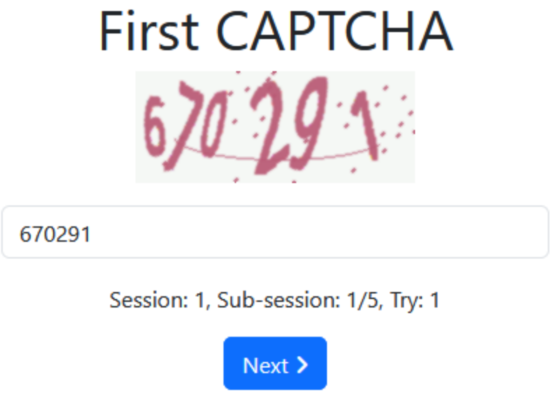
3.1. Participants

The study involved 12 voluntary participants (7 males and 5 females), ranging in age from 20 to 28 years old ($M = 23.75$, $SD = 2.26$). Based on the results of a pre-experiment demographic questionnaire, the majority of participants had prior experience with CAPTCHA schemes, particularly the checkbox variant. While 5 participants expressed frustration with resolving CAPTCHA challenges, the remaining participants maintained a neutral stance. According to most participants, the primary purpose of CAPTCHA systems is to prevent spam and automated programs from accessing online platforms. Additionally, the participants reported that they were generally able to complete the CAPTCHA tasks within 5 to 10 seconds.

3.2. Apparatus

The experiment participants used an Acer Aspire E5-575G laptop with the following hardware specifications:

- Processor: Intel(R) Core(TM) i7-6500U
- RAM: 16 GB
- OS: Windows 10 Home
- Display: 15.6 inch with 1920x1080 resolution
- Keyboard: Italian QWERTY layout
- Mouse: a generic external optical mouse (15 x 12 x 0,1 cm, 1000 DPI).


First CAPTCHA

670291

Session: 1, Sub-session: 1/5, Try: 1

Figure 1: A screenshot of the application interface depicting a user successfully completing a text-based CAPTCHA challenge

The researchers used this laptop to run the software for all participants, with audio volume and screen brightness both set to the maximum levels. The software was developed as a website using the Flask framework for the back-end with several packages [16–18], which was hosted locally on the device. The front end of the website was created using HTML and Bootstrap. Figure 1 shows a screen of the developed application.

Participants on the landing page selected an option from a dropdown menu according to an identifier assigned to them before the experiment. The website then automatically executed the appropriate predefined set of CAPTCHA challenges. Upon completion of the experiment, the website recorded the log data.

3.3. Procedure

Before the experiment, participants were asked to complete a questionnaire in order to gather demographic information, like full name, age, gender, country and other general questions regarding their previous experience with CAPTCHA schemes.

We proceeded to explain the purpose of this user study and then instructed the participants on how to go through the experiment. Afterwards, we gave them an identifier (a number). Sessions for each participant have been preconfigured based on their identifier, which represents the configuration that they will have to solve (selectable on the landing page of the website). The task given to participants was to complete a sequence of CAPTCHAs. All participants completed 12 sessions, which were composed of two CAPTCHA schemes chosen between:

	1	2	3	4	5	6	7	8	9	10	11	12
P1	TA	TC	TP	AT	AC	AP	CT	CA	CP	PT	PA	PC
P2	AT	AC	AP	CT	CA	CP	PT	PA	PC	TA	TC	TP
P3	CT	CA	CP	PT	PA	PC	TA	TC	TP	AT	AC	AP
P4	PT	PA	PC	TA	TC	TP	AT	AC	AP	CT	CA	CP
P5	TC	TP	TA	AC	AP	AT	CA	CP	CT	PA	PC	PT
P6	AC	AP	AT	CA	CP	CT	PA	PC	PT	TC	TP	TA
P7	CA	CP	CT	PA	PC	PT	TC	TP	TA	AC	AP	AT
P8	PA	PC	PT	TC	TP	TA	AC	AP	AT	CA	CP	CT
P9	TP	TA	TC	AP	AT	AC	CP	CT	CA	PC	PT	PA
P10	AP	AT	AC	CP	CT	CA	PC	PT	PA	TP	TA	TC
P11	CP	CT	CA	PC	PT	PA	TP	TA	TC	AP	AT	AC
P12	PC	PT	PA	TP	TA	TC	AP	AT	AC	CP	CT	CA

Table 1: Counterbalancing scheme used in the user study. (T = Text, A = Audio, P = Puzzle, C = Checkbox)

- Text: transcription of the text written inside a distorted image;
- Audio: transcription of what it is listened in a distorted audio;
- Puzzle: use a slider to complete the image with the missing piece;
- Checkbox: tick a checkbox to prove you are not a robot.

The two schemes are always repeated 5 times to complete a session. When the participant fails a scheme, it is proposed (a new one is generated, but of the same type) for a maximum of 5 trials. Participants also took a one-minute break every 3 sessions. We counterbalanced the experiment by randomly assigning a different order mode to each participant and arranging the sessions according to the order shown in Table 1.

For this study, we define:

- **Single-factor CAPTCHA**: The first CAPTCHA challenge that a participant completes within a session of trials.
- **Multi-factor CAPTCHA**: The pair of CAPTCHA challenges that a participant must complete within a session of trials.

During the experiment, we recorded the result (both correct and wrong) and the time spent on every CAPTCHA trial. Furthermore, we recorded the time for each of the 12 sessions and the time to complete the whole experiment (namely, from the beginning of the absolute first scheme of 1st session until the absolute last scheme of the 12th session).

After the experiment, participants were asked to complete a System Usability Scale [10] (SUS) questionnaire to evaluate multi-factor schemes and a questionnaire related to personal opinions to gather feedback.

3.4. Design

The experiment was a two-factor within-subject design. The two factors were:

- CAPTCHA type (4 levels):
  - Text
  - Audio
  - Puzzle
  - Checkbox
- CAPTCHA modes (2 levels):
  - Single-factor
  - Multi-factor

The dependent variables were the accuracy (we measured the error rate percentage), the completion time and the rating (a 1 to 5 score assigned to each combination of multi-factor CAPTCHA). To counterbalance, we arranged the sessions with a specific flow of execution as illustrated in the Procedure section (Table 1).

4. Results

All participants completed the experiment. For each participant, the experiment lasted about 25 minutes, including the introductory session. We tested significance using an analysis of repeated variance measures (ANOVA) [7]. We also analyzed data for significance using Scheffé, Bonferroni-Dunn and Fisher LSD Post Hoc Comparison tests.

4.1. Completion time

Regarding completion time, the entire experiment took an average of 28:56 minutes ($S.D.$ 3:26 minutes). The chart in Figure 3 shows the average times obtained by the participants for each CAPTCHA pair (multi-factor), while the chart in Figure 5 shows the times obtained by the participants in solving single CAPTCHAs (single-factor). The times of the single-factor mode were selected as the times recorded by the users in solving the first CAPTCHA of the two provided in the multi-factor mode; this choice is explained by the fact that in the multi-factor resolution, the CAPTCHAs are proposed sequentially, and therefore the second CAPTCHA is accessed only after solving the first one. The solution time of the first CAPTCHA, therefore, can also be considered as single-factor resolution time. However, again the chart in Figure 5 shows that there is no significant difference between solving a given CAPTCHA first or second.

The participants performed most rapidly on the CAPTCHA pair $(Checkbox, Puzzle)$ requiring $12.16s$ for CP and $12.05s$ for PC. They were slightly slower on the pair $(Puzzle, Text)$ taking $13.89s$ for TP and $16.81s$ for PT. In contrast, the CAPTCHA pair $(Text, Audio)$ demanded the longest completion time of $33.22s$ and exhibited the slowest performance. These findings suggest that certain CAPTCHA combinations, such as the pairing of Checkbox

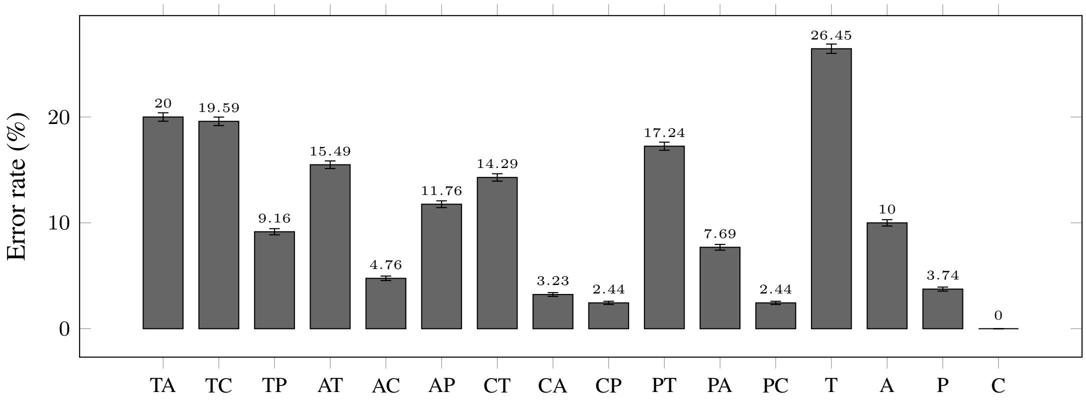


Figure 2: Error rate of all CAPTCHA sequences.
T = Text, A = Audio, P = Puzzle, C = Checkbox.

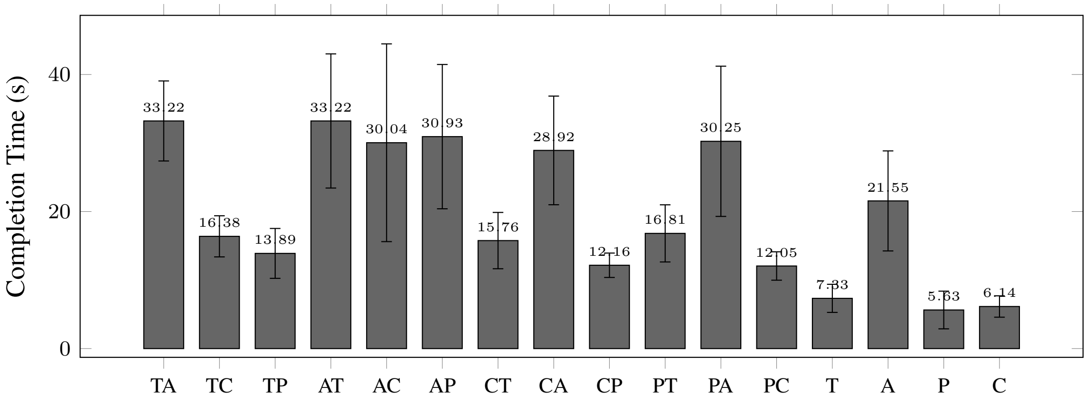


Figure 3: Completion time of all CAPTCHA sequences.
T = Text, A = Audio, P = Puzzle, C = Checkbox.

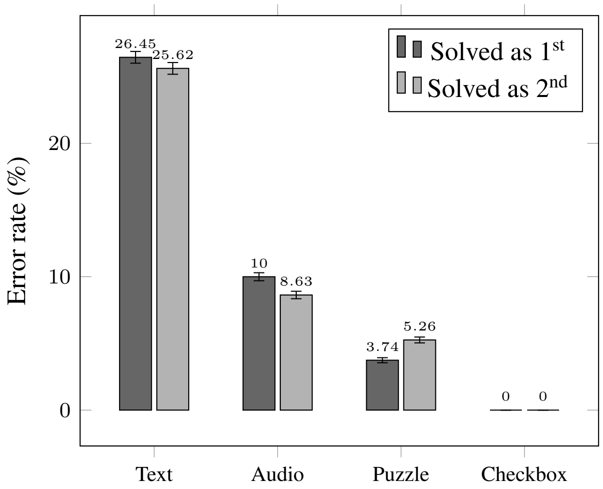


Figure 4: Error rate when the scheme was solved as first or as second.

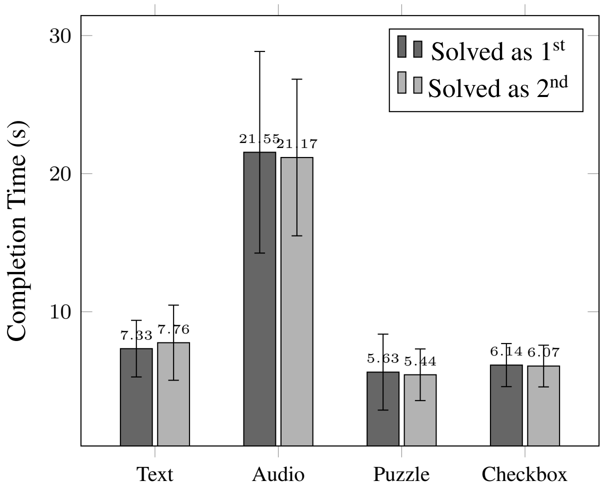


Figure 5: Time to complete the scheme when solved as first or as second.

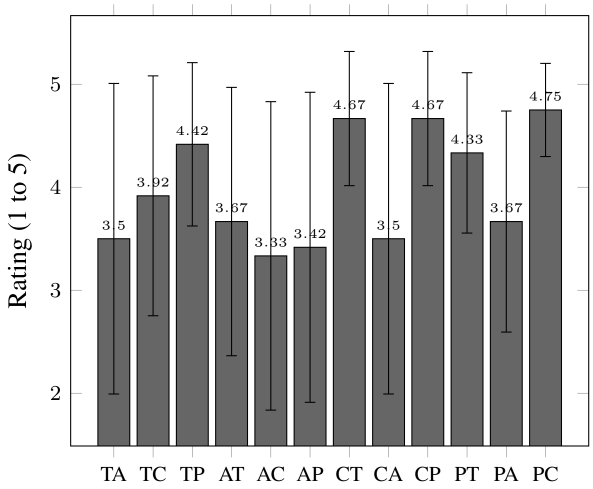


Figure 6: Rating of multi-factor CAPTCHA sequences.
T = Text, A = Audio, P = Puzzle, C = Checkbox.

and Puzzle, may optimize efficiency by minimizing the time required for users to successfully verify their identity.

The ANOVA results showed that the effect of CAPTCHA type on completion time was statistically significant ($F_{3,33}$ = 35.081, $p < .0001$). The effect of mode on completion time was statistically significant ($F_{1,11}$ = 589.942, $p < .0001$). The CAPTCHA type × mode interaction effect was statistically significant ($F_{3,33}$) = 10.019, $p < .0001$). Table 2 also shows significance for Scheffé, Bonferroni-Dunn and Fisher LSD Post Hoc Comparison tests.

4.2. Error rate

Regarding the number of errors, Figure 2 shows the average error rate obtained by the participants for each CAPTCHA pair (multi-factor). In contrast, Figure 4 shows the average error rate obtained by the participants in solving single CAPTCHAs (single-factor). For the measuring of the error rate for single-factor are valid the same arguments are shown in Section 4.1.

The participants exhibited a lower error rate of 2.44% when interacting with the $(Checkbox, Puzzle)$ CAPTCHA pair. In contrast, textual CAPTCHA schemes were found to be the most error-prone, as evidenced by the higher overall error rate whenever they were included in the multi-factor CAPTCHA sequences. The data suggests that the $(Checkbox, Puzzle)$ CAPTCHA pair was more effectively solved by the participants, resulting in a substantially lower error rate of 2.44% compared to the text-based CAPTCHA challenges. In the last analysis, Figure 4 indicated that the order in which the CAPTCHA was solved did not have a statistically significant effect on the error rate.

Pair		Scheffé	Bonferroni-Dunn	Fisher LSD
ST	SA	*	*	*
ST	MA	*	*	*
ST	SP	*	*	*
ST	MP	*	*	*
ST	SC	*	*	*
ST	MC	*	*	*
MT	SA	*	*	*
MT	MA	*	*	*
MT	SP	*	*	*
MT	MP	*	*	*
MT	SC	*	*	*
MT	MC	*	*	*
SA	SC		*	*
SA	MC		*	*
MA	SC		*	*
MA	MC		*	*

Table 2: Statistical Significance Tests for the Completion Time: Scheffé and Bonferroni-Dunn. Non-significant pairwise comparisons were omitted.
ST = Single-factor Text, MT = Multi-factor Text,
SA = Single-factor Audio, MA = Multi-factor Audio,
SP = Single-factor Puzzle, MP = Multi-factor Puzzle,
SC = Single-factor Checkbox, MC = Multi-factor Checkbox

The ANOVA results showed that the effect of CAPTCHA type on error rate was statistically significant ($F_{3,33}$ = 32.632, $p < .0001$). The effect of mode on error rate was not statistically significant ($F_{1,11}$ = 0.137, ns). The CAPTCHA type × mode interaction effect was not statistically significant ($F_{3,33}$ = 0.082, ns). Table 3 shows significance for Scheffé, Bonferroni-Dunn and Fisher LSD Post Hoc Comparison tests.

4.3. Rating

The results presented in Figure 6 indicate that participants provided good ratings for the various CAPTCHA challenge pairs evaluated in the study. The rating scale ranged from 0 to 5 stars, with 5 representing the highest level of satisfaction and 0 the lowest. Overall, the CAPTCHA challenge pairs received average ratings exceeding 3.5 out of 5, suggesting a generally positive response from the participants. The CAPTCHA challenge pair that received the highest rating from participants was $(Puzzle, Checkbox)$ with a score of 4.75 out of 5, followed by $(Checkbox, Text)$ and $(Checkbox, Puzzle)$. In contrast, the CAPTCHA pairs that incorporated an audio-based challenge received lower ratings from the participants, with the pair that utilized the audio CAPTCHA obtaining the lowest score of 3.33 out of 5. This suggests that the audio-based CAPTCHA schemes were less preferred by the par-

Pair		Scheffé	Bonferroni-Dunn	Fisher LSD
ST	SA	*	*	*
ST	MA	*	*	*
ST	SP	*	*	*
ST	MP	*	*	*
ST	SC	*	*	*
ST	MC	*	*	*
MT	SA	*	*	*
MT	MA	*	*	*
MT	SP	*	*	*
MT	SP	*	*	*
MT	SC	*	*	*
MT	MC	*	*	*
SA	SC			*
SA	MC			*
MA	SC			*
MA	MC			*

Table 3: Statistical Significance Tests for the Error rate: Scheffé and Bonferroni-Dunn. Non-significant pairwise comparisons were omitted.
ST = Single-factor Text, MT = Multi-factor Text,
SA = Single-factor Audio, MA = Multi-factor Audio,
SP = Single-factor Puzzle, MP = Multi-factor Puzzle,
SC = Single-factor Checkbox, MC = Multi-factor Checkbox

Pair		Scheffé	Bonferroni-Dunn	Fisher LSD
ST	SA	*	*	*
ST	MA			*
MT	SA			*
MT	SC			*
SA	SP	*	*	*
SA	MP		*	*
SA	SC	*	*	*
SA	MC			*
MA	SP			*
MA	SC		*	*

Table 4: Statistical Significance Tests for the user rating: Scheffé and Bonferroni-Dunn. Non-significant pairwise comparisons were omitted
ST = Single-factor Text, MT = Multi-factor Text,
SA = Single-factor Audio, MA = Multi-factor Audio,
SP = Single-factor Puzzle, MP = Multi-factor Puzzle,
SC = Single-factor Checkbox, MC = Multi-factor Checkbox

	1st	2nd	3rd	4th
Text	1	3	8	-
Audio	-	2	1	9
Puzzle	3	4	2	3
Checkbox	6	5	1	-

Table 5: Ranking of the four CAPTCHA schemes.

ticipants compared to the other CAPTCHA types evaluated in the study.

Regarding the single-factor CAPTCHA schemes, Table 5 presents the ranking provided by the participants. The data indicates that the Checkbox CAPTCHA was the most preferred, receiving 6 votes for the top position. This was followed by the Puzzle CAPTCHA, the Text CAPTCHA, and the Audio CAPTCHA in descending order of participant preference. The higher preference for the Checkbox CAPTCHA suggests that it was perceived as the most user-friendly and effective among the single-factor CAPTCHA schemes tested in the study.

The ANOVA results showed that the effect of CAPTCHA type on the rating was statistically significant ($F_{3,33}$ = 12.787, $p < .0001$). The effect of mode on rating was not statistically significant ($F_{1,11}$ = 2.555, $p > .05$). The CAPTCHA type $\times$ mode interaction effect was statistically significant ($F_{3,33}$ = 7.121, $p < .001$). Table 4 also shows significance for Scheffé, Bonferroni-Dunn and Fisher LSD Post Hoc Comparison tests.

4.4. User Satisfaction and Free-form comments

At the end of the experiments, all involved people filled out a SUS questionnaire to measure participant satisfaction. Half of the participants expressed a preference for the single-factor Checkbox scheme (6 out of 12 ranked it first compared to the others), while the least liked was the single-factor Audio scheme (ranked last 9 times out of 12). The most liked multi-factor combination was Checkbox-Puzzle, followed by Puzzle-Checkbox. On the other hand, the least liked was the Text-Audio combination. Overall, the SUS questionnaire regarding the combination of CAPTCHAs in multi-factor mode had the following average score: 76.88 (SD = 13.15) which is a good value. Table 5 shows participants' preference about individual CAPTCHA modes (1^{st} to 4^{th} positions).

5. Discussion

The data collected showed that the textual CAPTCHA had the highest error rate among the various CAPTCHA's tested (26%) while for other CAPTCHA's the participants made fewer errors (and, for the checkbox-type CAPTCHA, no errors were detected given the nature of the CAPTCHA to analyse the movement of the mouse to discriminate whether the user is human or not). The influence of the errors made with the textual CAPTCHA was also propagated in the multimodal CAPTCHA, as can be seen in Figure 2. Figures 4 and 5, however, show that there was no correlation between the number of errors made and the order in which the textual CAPTCHA occurred: in general, for no type of CAPTCHA was this observed.

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th
Text-Checkbox	1		2	2		2	1	3	1			
Text-Audio	2				1	2		1			2	4
Text-Puzzle		1	2	4	2				1	1	1	
Audio-Checkbox		1		1	1	1	1	1		3		3
Audio-Text		1		1	1		3	1		1	2	2
Audio-Puzzle				1		1	3	1	1	3	2	
Puzzle-Text		1	6	2					1	1		1
Puzzle-Checkbox	3	4			1		1	1			2	
Puzzle-Audio						3	2	2	2		2	1
Checkbox-Audio			1		2	1	1	1	5	1		
Checkbox-Text	2	2	1	1	3	1		1		1		
Checkbox-Puzzle	4	2			1	1			1	1	1	1

Table 6: Rank of the twelve CAPTCHA sequences.

Also concerning time, no correlation was observed between the time required and the order in which the individual CAPTCHA was to be completed, as it is possible to see in Figure 3. The audio-type CAPTCHA, on the other hand, proved to be the slowest (with an average of about 21 seconds) compared to the 5-7 seconds required by the other CAPTCHA; this is because the user has to wait for the time of the pronunciation (interspersed with a brief but random noise time). It must be considered that although it was the most 'demanding' in terms of time, this type of CAPTCHA is particularly important for those who have difficulties with textual CAPTCHA. The study participants did not exhibit any apparent visual or auditory impairments. However, the inclusion of individuals with disabilities or specific learning disorders could potentially alter the results of the experiment. Nevertheless, this study should be considered a valid experiment, and the researchers plan to address this aspect as part of their future work.

In the analysis of the multi-modal pairs, the CAPTCHA puzzle and checkbox pair (regardless of the order) were the fastest and also those with the fewest errors made, making it the seemingly best pair in the multi-factor CAPTCHA.

In general, participants seemed to have liked all types of pairs, with average ratings of more than 3 out of 5. Among the most liked pairs is the puzzle, checkbox pair (with 4.6-4.7 out of 5). Surprisingly, the text-puzzle pair was also highly liked (average 4.3/5) despite taking longer (14-16 seconds) and involving more errors (9-17).

Analyses on some pairs which showed substantial differences in the order for the number of errors (t-c 19, c-t 14 or even t-p 9 and p-t 17) will be the subject of future studies: apparently, the order did not generate particularly discordant results but since for these pairs it did happen we will look for the reasons for this.

The security of multi-factor CAPTCHAs was not comprehensively tested in this study. The primary focus of the research was to evaluate user interaction and experiences with multi-factor CAPTCHA schemes, rather than directly assessing their security against automated attacks. While this limitation is acknowledged, the researchers suggest that future work could explore the security aspects of multi-factor CAPTCHAs in more depth, potentially by incorporating tests with automated tools to assess their resilience against bots and other malicious programs.

6. Conclusions and Further Works

Since the advent of CAPTCHAs, attempts have been made to introduce a multi-factor CAPTCHA design in order to improve security and prevent bots from automatically submitting web forms. The current study investigated the introduction of multi-factor CAPTCHA designs as a means to enhance security and mitigate the threat of automated form submissions. Specifically, the accuracy and error rates of four distinct CAPTCHA types were evaluated, comparing their performance when employed as single-factor challenges versus integrated into multi-factor verification schemes. Participants engaged in a series of 12 counterbalanced sessions to assess these different modalities. Given that CAPTCHAs must strike a balance between security and minimizing complexity or user frustration, the researchers also analyzed participants' preferences by soliciting evaluations after each CAPTCHA pair as well as an overall assessment after the experiment.

To conduct this user study, we developed a Flask-based web application that allowed participants to solve different types of CAPTCHAs, following a controlled procedure. The application recorded user performance metrics in real-time, their feedback both in terms of votes and free comments.

The analysis of the results revealed that the audio-based CAPTCHA was the least preferred option among participants, as it was the slowest to complete, despite being one of the most accurate. In contrast, the checkbox CAPTCHA demonstrated the highest accuracy, while the puzzle CAPTCHA was the fastest scheme. Regarding the multi-factor CAPTCHA modes, the sequences combining the puzzle and checkbox challenges, namely Puzzle-Checkbox and Checkbox-Puzzle, were found to be the most efficient in terms of the time required by users and the error rate. Conversely, the Text-Audio sequence was deemed the least efficient.

Several opportunities exist to extend this research, addressing the limitations outlined in section 5. First, the experiment should be replicated with a larger participant pool to confirm the findings and investigate the interaction between CAPTCHA type and mode (single-factor vs. multi-factor). Additionally, the researchers intend to explore additional CAPTCHA modalities, such as CAPTCHAs with voice recognition for medical forms [4], and their multi-factor combinations, increasing the number of factors. Moreover, the researchers will examine the implications of employing multi-factor CAPTCHAs in educational games [3, 5] to ensure the participation of human players .

Furthermore, a comprehensive experiment will be conducted to assess the security of multi-factor CAPTCHAs, including evaluating the effectiveness of recently released neural models. Finally, we intend to analyze the use of multi-factor CAPTCHAs with participants who have disabilities affecting their vision, and hearing or have specific learning disorders, such as dyslexia.

References

[1] A. Algwil. A security analysis of text-based captcha schemes. 2:309–323, 09 2023.

[2] A. S. Almazyad, Y. Ahmad, and S. A. Kouchay. Multimodal captcha: A user verification scheme. *International Journal of Engineering Research & Technology*, 2012.

[3] G. Coppola, G. Costagliola, M. De Rosa, and V. Fuccella. Domus: A multi-user tui game for multi-touch tables. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '20, New York, NY, USA, 2020. Association for Computing Machinery.

[4] E. Corbisiero, G. Costagliola, M. De Rosa, V. Fuccella, A. Piscitelli, and P. Tabari. Speech recognition in healthcare: A comparison of different speech recognition input interactions. In Y.-W. Chen, S. Tanaka, R. J. Howlett, and L. C. Jain, editors, *Innovation in Medicine and Healthcare*, pages 142–152, Singapore, 2023. Springer Nature Singapore.

[5] G. Costagliola, M. De Rosa, V. Fuccella, A. Piscitelli, and P. Tabari. Domus: An educational multiplayer game for touch tables using a tangible user interface. In P. Zaphiris and A. Ioannou, editors, *Learning and Collaboration Technologies*, pages 3–16, Cham, 2024. Springer Nature Switzerland.

[6] Dayanand, M. Saloni, and W. Jeberson. Multi-layered captcha - a new approach to tackle web robots. *International Journal of Engineering Research & Technology*, 2018.

[7] E. Girden. *ANOVA: Repeated Measures*. Number No. 84 in ANOVA: Repeated Measures. SAGE Publications, 1992.

[8] M. Guerar, L. Verderame, M. Migliardi, F. Palmieri, and A. Merlo. Gotta captcha 'em all: A survey of 20 years of the human-or-computer dilemma. *ACM Computing Surveys*, 2021.

[9] E. Igbekele, A. A. Adebiyi, F. Ibikunlem, and M. Adebiyi. Research trends on captcha: A systematic literature. *International Journal of Electrical and Computer Engineering*, 2021.

[10] B. John. Sus: a "quick and dirty" usability scale. *Usability Eval. Ind*, 189, 1995.

[11] M. Moradi and M. Keyvanpour. Captcha and its alternatives: A review. *Security and Communication Networks*, 2014.

[12] U. P, Member, IAENG, S. K, and R. P. Smart captcha to provide high security against bots. *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering*, 2019.

[13] D. U. Pujeri and D. P. S. Aithal. Smart captcha version 1. *GIS SCIENCE JOURNAL*, 2021.

[14] S. Sharma and D. Singh. Captcha in web security and deep-captcha configuration based on machine learning. In *2024 3rd International Conference for Innovation in Technology (INOCON)*, pages 1–6, 2024.

[15] S. Vikram, Y. Fan, and G. Gu. Semage: A new image-based two-factor captcha. *ACSAC '11: Proceedings of the 27th Annual Computer Security Applications Conference*, 2011.

[16] `https://pypi.org/project/captcha/`

[17] `https://pypi.org/project/flask/`

[18] `https://pypi.org/project/flask-Captchaify/`

MVP: Mind Map-Based Video Searching Platform

[1]Taeghyun Kang, [2]Hyungbae Park, [3]Sunae Shin

[1]University of Central Missouri, Warrensburg, MO, USA
[2]University of North Georgia, Dahlonega, GA, USA
[3]Georgia Gwinnett College, Lawrenceville, GA, USA
Email: tkang@ucmo.edu, hpark@ung.edu, sshin7@ggc.edu

Abstract

Video has now surpassed all other types of content as the most common medium of disseminating information. Many videos are created for entertainment, but more are being made for educational or instructional purposes. Users can now easily create videos using platforms such as YouTube, TikTok, and Instagram, which have search algorithms that recommend videos to users based on their interests. But with numerous videos being produced on these sites, finding videos which provide the information they need has become increasingly difficult for users. In this paper, we have built a video search platform using a mind map architecture. Mind map is an effective and efficient tool and a valuable method for improving critical thinking. It helps people to start exploring and expanding the topics with the natural way of doing things in the brain. We have developed MVP (Mind Map-Based Video Searching Platform), a service that allows users to build mind maps and upload videos to each node within the mind map.

Index terms— Video sharing platform, Video searching platform, Mind map, UX design

1 Introduction

The speed of the Internet has improved dramatically as a result of the growth of Internet-based technologies and networks, and video content has emerged as the most significant information distribution medium. Individuals can now easily produce videos thanks to advancements in online video sharing and production platforms. As more experts in different fields produce videos about content relevant to their fields of work, the importance of the knowledge found in the videos has increased significantly. Users can search for videos by using a tag or video title, and platforms often provide a service that uses a complex algorithm to suggest videos that users may be interested in. Nevertheless, the countless videos created by many users every day make it difficult to perform searches. In addition, it is even more challenging to find a specific piece of information within a video. Two main factors identified on Wikipedia could help address this issue. To begin, Wikipedia provides a table of contents that provides an overview of the topics that the user has searched for. This table of contents allows users to quickly navigate to topics of interest, in a similar fashion to a car's navigation system that guides users to their destinations. If users have some background knowledge of the subject they are looking for, the information can be easily found by means of links in the content table instead of reading the complete text. Secondly, links in the text in Wikipedia provide information that is either directly or indirectly relevant to the subject. These links, like the recommendation system of an online video-sharing site, can often pique users' interest in exploring additional details. Figure 1 shows a guide to becoming a front-end developer using a mind map structure [1]. However, since the nodes of the mind map are not directly linked to their detailed information, users must go through the hassle of searching for additional information for each node.

In this paper, we have developed an online sharing and searching platform with an architecture of mind maps. Rather than creating an effective video searching algorithm, we focused on adopting a mind map tree structure to develop a well-defined repository architecture to store videos. The remainder of the paper is structured as follows: Section 2 describes the motivation of the paper and reviews

[0]DOI reference number: 10.18293/DMSVIVA2024-004

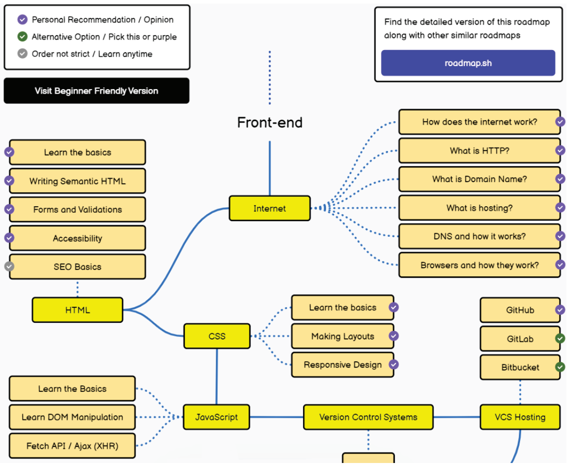


Figure 1: Roadmap of a Frontend Developer Using a Mind Map Structure

context information and literature references. Section 3 describes requirements and functionalities of the developed platform. Section 4 depicts the platform's implementation as a proof-of-concept and as a basis for further research to demonstrate the platform's efficacy. Section 5 concludes the paper.

2 Related work

As online video-sharing platforms are getting more attention and popularity, the number of videos posted online has been exponentially increased. Online video-sharing platforms are utilizing keyword tagging and searching for users to find videos they are interested in. However, the extreme growth and distribution of video on the Internet has made searching for specific videos that satisfy users' expectation using simple text and keyword-based queries a significant challenge. These existing indexing, tagging, and searching has a limited capability as computer AI's image recognition and linguistic analysis still have some limitation to perfectly understand images on videos and human language. In order to efficiently and effectively retrieve relevant videos from the enormously large video database, various techniques have been developed and implemented.

Temporal Localization with Natural Language: Hendricks et al. [5], Shao et al. [19], Zhang et al. [23], and Zhao et al. [24] addressed the problems of retrieving a temporary segment from a video for a given natural language query. These schemes don't require any pre- or post-processing of the video for effective moment localization. This is a challenging problem as each moment may have different semantics. Learning discriminative features out of videos requires an ability to deal with its flexibility and complexity of moment description. They mainly focus on retrieving a specific temporal segment from a video, but the feature could be simply extended to video retrieval systems.

Video-Text Embedding Models: Miech et al. [17] proposed a model that learns video-text embeddings from heterogeneous data sources. Similarly, Mithun et al. [18] and Liu et al. [16] proposed a novel framework that utilizes available multi-modal cues from videos for the cross-modal video-text retrieval task. For effective video retrieval, [18] used a fusion strategy and those multi-modal features include different visual characteristics, audio inputs, and text. They also proposed a new loss function that further exploits the multimodal correlation. [16] framework effectively uses embeddings from different scene, objects, actions, etc. by learning their combination in order to render them more discriminative. The framework retrieves video contents using a free-form text query that may contain both general and specific information.

Content-Based Video Retrieval (CBVR): In order to tackle the limitations of keyword-based video search, content-based video retrieval techniques have been proposed [10, 20, 9, 15]. CBVR systems can retrieve a list of videos by various types of queries such as query by objects, query by location, query by example, query by sketch, query by natural language, etc. In this paper, we propose a simple but innovative way that utilizes a mind map [7, 8] for efficient and effective video retrieval search engine. Mind maps have been adopted and used in various ways such as a learning strategy [21, 14, 25], a user modeling and recommender system [6, 11], document summarization [22, 13], etc. Our proposed platform use a mind map in a way that mimics users' flow of conscious when they need to search certain videos. This paper discusses the limitations of the current video information retrieval systems, a new way to use a mind map, and the technical requirements for implementing a flow of human consciousness-based search engine. To the best of our knowledge, our work is the first to implement a video search platform using a mind map and to tackle and overcome challenges and issues in implementing the platform. Our platform allows users to organize (index and search) videos based on the flow of human consciousness (thinking process) via mind maps.

3 The Functionalities and UX Design of the MVP

To design an effective and efficient video search platform, we first elicited user and external interface requirements. The majority of video search sites, such as YouTube, provide search and recommendations based on the user's viewing and search history. These platforms are concentrating on improving their search and recommendation algorithms. However, a user cannot figure out whether the video includes the information the user needs until he/she watches the entire video. According to the statistic [2], users tend to watch short videos, and shorter videos are particularly engaging when they offer educational and insightful content [3]. The details found in the video becomes easier to locate and clearer as the video's duration is reduced. The users will not always be able to specify the information they need. Even if the user searches based on incomplete information, the platform must be able to navigate to the information the user ultimately wants, like the navigation system in a car. The user requirements are summarized in Table 1.

Table 1: User and External Interface Requirements

1. The platform should provide videos that deliver knowledge for educational purposes.
2. Users can easily search for information contained in videos (not the videos themselves).
3. The mind map ecosystem should be managed and enhanced through collective intelligence, similar to Wikipedia.
4. The platform is able to recommend the videos to users based on the relationship between the information in the video. When user watch informative videos like tutorial, knowledge guide, learning, and academic classes videos, it should not recommend video based on the user or other user's search history.

We propose MVP (Mind Map-Based Video Searching Platform) that implements the requirements listed above. The tree-like structure of a mind map is suitable for representing knowledge structured in a hierarchical manner, such as the table of contents in Wikipedia. In MVP, a video should be divided and saved according to the mind map's hierarchical detail. It allows users to search for information when moving through the mind map's hierarchical structure, allowing them to easily locate the information they need without having to use complicated search algorithms. The subject that users are investigating appears as a root node in the mind map, but it may actually be a sub-topic of another piece of material.

3.1 UX design of MVP

When users search for videos, instead of searching for individual videos, the platform displays an information ecosystem (Mind Map) that contains the desired information through a mind map. To meet the requirements defined in Table 1, The interface is designed as shown in Figure 2 and the functionalities of the mind map is summarized in Table 2.

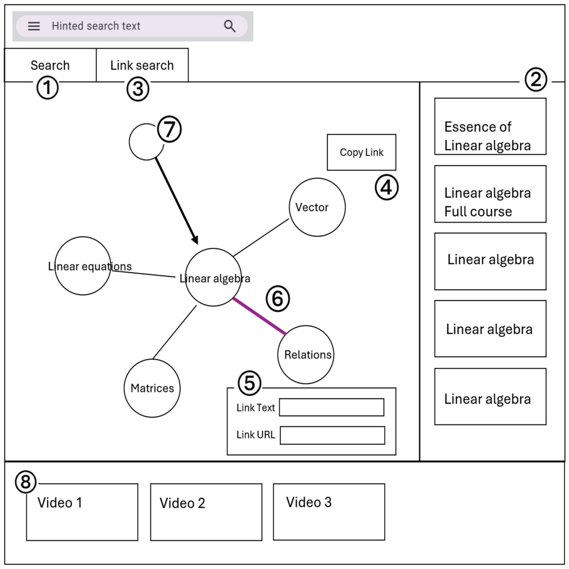


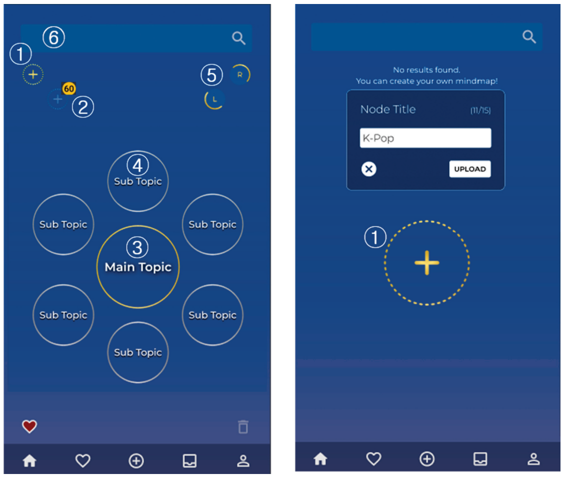
Figure 2: UX Design Diagram of the Mind Map

Table 2: Functionalities of the Mind Map

Number in Figure 2	Description
1	Search mind maps created by individual users.
2	The search results are listed in Section 2.
3	The layout of the Search tab and the Link Search tab is identical. The Search tab is used to find and extract link information for related mind maps when connecting a node in the current mind map to others.
4	When connecting related nodes from other mind maps, it extracts link information using the ”Copy Link” button.
5	Users can use the extracted link information to create new sub-nodes in the mind map they are currently using.
6	When users move nodes in the mind map to find desired information, the frequency of movement is differentiated by the color and thickness of the lines.
7	User search for other mind maps that reference the current mind map.
8	When a user clicks a node in the mind map, the videos associated with that node are listed in Section 8.

Smartphones have overtaken as the most popular device for watching videos. According to a statistic, mobile devices account for 70% of overall YouTube watch time [4].

We have designed and built a video search platform using a mind map architecture for mobile devices. The number of sub-nodes shown on the screen is limited to 10 due to the smartphone's screen size, and sub-nodes with more than 10 are not shown on the screen but run in the background. However, the user can navigate through hidden sub-nodes by rotating the sub-nodes left and right (placed at #5 in Figure 3). Also, the buttons that create the root node and the sub-nodes have been separated and positioned in the upper left corner to avoid unintended actions that may occur due to incorrect screen touch.


No results found.
You can create your own mindmap!

Node Title (11/15)

K-Pop

Figure 3: UX Design of the Mind Map in MVP

Functionalities of the mind map in MVP is summarized in Table 3.

Table 3: Functionalities of the Mind Map for Mobile Device

Number in Figure 3	Description
1	Create a root node of a mind map
2	Create a sub-node of a mind map
3	Root node
4	Sub-node
5	Rotate sub-nodes around the root node
6	Search a mind map

3.2 Searching Process of MVP

The user must create a mind map that functions as table of contents. If other users have already created a mind map for the same topic, the user can either upload a video using one of the existing mind maps crreated by others or create a new mind map of their own. Users can generate multiple mind maps for a single topic using the MVP. Even if the main topic is the same, the way the sub-contents are arranged and organized will differ. As a result, when users are looking for a specific mind map, the structure of the mind map can be used as a source of information. For example, in Figure 4, JPA (Java Persistence API) is a collection of classes and method that allows easy interaction with database instance. When a user searches for a mind map using the keyword "JPA", they can navigate all mind maps that contain a "JPA" node. In mind maps where JPA node is used as the root, the user can get an overview of the details needed to learn JPA from the structure of the mind map. If the "JPA" node is a sub-node in a mind map, the user can explore how JPA relates to other technologies and topics.

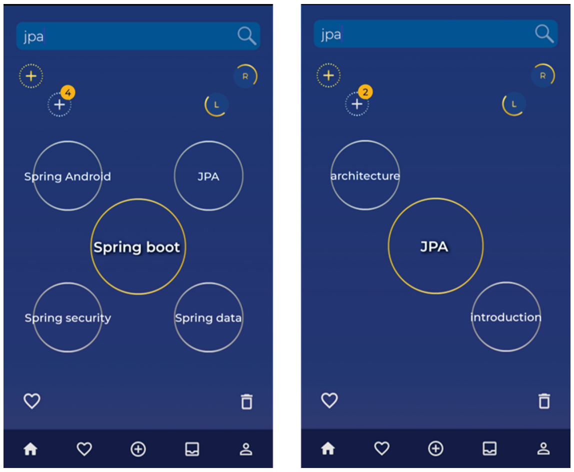


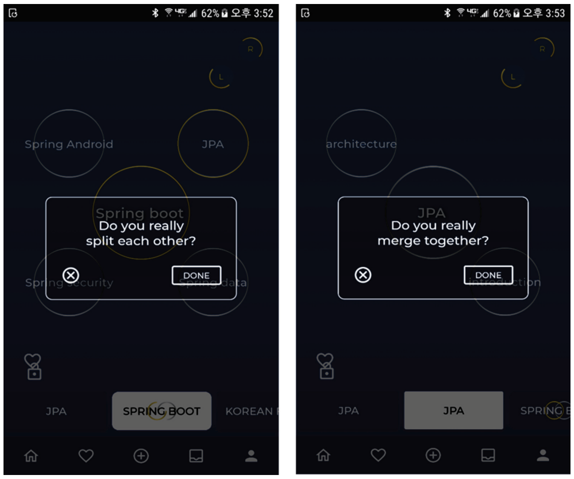
Figure 4: Mind Map Search Outcome of MVP

3.3 Mind Map Integration and Division

Mind maps help users to see the entire picture by connecting one topic to the next. Users can create their own knowledge map that shows where a video can be found. To compose a refined mind map, MVP provides a function for dividing or integrating the mind maps as shown in Figure 5.

4 Implementation

To collect data to verify the efficiency of MVP as a video sharing platform, The app is implemented using android with Kotlin and server-side APIs are developed using PHP. The MVP has restricted video length to 5 minutes to ensure that each video contains only the essential information [12]. Additionally, dividing the content into smaller nodes within a mind map helps users locate information more quickly and effectively. The operating environments are summarized in



Do you really split each other?

Do you really merge together?

Figure 5: Integration and Division of Mind Map in MVP

the Table 4. Figure 6 shows the entity diagram for creating a mind map. Every node has a unique id, and, if a node is a sub-node in the mind map, it can be identified with the parent node and depth of the node in the MindMap table. When a video is uploaded to Amazon Cloud Storage, it records the file name, title, description of the video, URL where a video is stored, etc.

Table 4: Operating Environments of MVP

Amazon Server	Description
EC2	Hosting server
ELB	Elastic Load Balancing
RDS1	MySQL 8.0.15 Community, Master databse
RDS2	MySQL 8.0.15 Community, Read replicas
S3	Secure cloud storage for videos
ElasticCache	Session server, cache

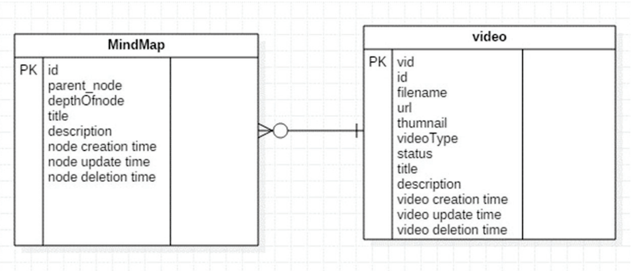


Figure 6: Entity Diagram for Mind Map in MVP

5 Conclusion

For the digital generation, videos are a more interactive medium. Owing to the long-term COVID-19 pandemic, many people have become accustomed to taking classes online. So far, videos for basic amusement have dominated the mainstream online video-creation platforms. The demand for educational and insightful videos to obtain information, on the other hand, is also steadily rising. MVP (Mind Map-Based Video Searching Platform) was created to provide a more effective and efficient way to store and share videos with the help of mind maps. When users search for new content for learning or educational purposes, they often rely on keyword searches based on fragmented information. This can lead to loss of interest or difficulty in acquiring the desired information, especially if they lack sufficient background knowledge. In the age of information overload, mind maps can leverage collective intelligence to build a high-level video ecosystem. By showing information relevance during the search process, mind maps enable users to easily grasp the overall structure of the information, even when they only have fragmented details.

References

[1] Frontend developer. https://roadmap.sh/frontend.

[2] Average youtube video length as of december 2018, by category. https://www.statista.com/statistics/1026923/youtube-video-category-average-length/, 2018.

[3] Distribution of videos removed from youtube worldwide from 2nd quarter 2019 to 3rd quarter 2023, by reason. https://www.statista.com/statistics/1132956/share-removed-youtube-videos-worldwide-by-reason/, 2023.

[4] Youtube usage statistics. https://worldmetrics.org/youtube-usage-statistics/, 2024.

[5] L. Anne Hendricks, O. Wang, E. Shechtman, J. Sivic, T. Darrell, and B. Russell. Localizing moments in video with natural language. In *Proceedings of the IEEE international conference on computer vision*, pages 5803–5812, 2017.

[6] J. Beel. Towards effective research-paper recommender systems and user modeling based on mind maps. *arXiv preprint arXiv:1703.09109*, 2017.

[7] T. Buzan. *Make the most of your mind.* Simon and Schuster, 1984.

[8] T. Buzan and B. Buzan. *How to mind map.* Thorsons London, 2002.

[9] L. Cao, X.-M. Liu, W. Liu, R. Ji, and T. Huang. Localizing web videos using social images. *Information Sciences*, 302:122–131, 2015.

[10] S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong. A fully automated content-based video search engine supporting spatiotemporal queries. *IEEE transactions on circuits and systems for video technology*, 8(5):602–615, 1998.

[11] M. H. Dlab. Experiences in using educational recommender system elars to support e-learning. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 672–677. IEEE, 2017.

[12] P. J. Guo, J. Kim, and R. Rubin. How video production affects student engagement: An empirical study of mooc videos. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 41–50, 2014.

[13] G.-J. Hwang, M.-R. A. Chen, H.-Y. Sung, and M.-H. Lin. Effects of integrating a concept mapping-based summarization strategy into flipped learning on students' reading performances and perceptions in chinese courses. *British Journal of Educational Technology*, 50(5):2703–2719, 2019.

[14] D. Indriani and I. Mercuriani. The effectiveness of experiential learning model by using mind map to the understanding of concepts on fungi materials at the tenth-grade students of senior high school. In *Journal of Physics: Conference Series*, volume 1567, page 042081. IOP Publishing, 2020.

[15] Y.-G. Jiang, J. Wang, Q. Wang, W. Liu, and C.-W. Ngo. Hierarchical visualization of video search results for topic-based browsing. *IEEE Transactions on Multimedia*, 18(11):2161–2170, 2016.

[16] Y. Liu, S. Albanie, A. Nagrani, and A. Zisserman. Use what you have: Video retrieval using representations from collaborative experts. *arXiv preprint arXiv:1907.13487*, 2019.

[17] A. Miech, I. Laptev, and J. Sivic. Learning a text-video embedding from incomplete and heterogeneous data. *arXiv preprint arXiv:1804.02516*, 2018.

[18] N. C. Mithun, J. Li, F. Metze, and A. K. Roy-Chowdhury. Learning joint embedding with multimodal cues for cross-modal video-text retrieval. In *Proceedings of the 2018 ACM on international conference on multimedia retrieval*, pages 19–27, 2018.

[19] D. Shao, Y. Xiong, Y. Zhao, Q. Huang, Y. Qiao, and D. Lin. Find and focus: Retrieve and localize video events with natural language queries. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 200–216, 2018.

[20] C.-W. Su, H.-Y. M. Liao, H.-R. Tyan, C.-W. Lin, D.-Y. Chen, and K.-C. Fan. Motion flow-based video retrieval. *IEEE Transactions on Multimedia*, 9(6):1193–1201, 2007.

[21] A. Vimalaksha, S. Vinay, and N. Kumar. Hierarchical mind map generation from video lectures. In *2019 IEEE Tenth International Conference on Technology for Education (T4E)*, pages 110–113. IEEE, 2019.

[22] R. Yulianto and S. Mariyah. Building automatic mind map generator for natural disaster news in bahasa indonesia. In *2017 International Conference on Information Technology Systems and Innovation (ICITSI)*, pages 177–182. IEEE, 2017.

[23] S. Zhang, H. Peng, J. Fu, and J. Luo. Learning 2d temporal adjacent networks for moment localization with natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12870–12877, 2020.

[24] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2914–2923, 2017.

[25] S. Zubaıdah, N. M. Fuad, S. Mahanal, and E. Suarsını. Improving creative thinking skills of students through differentiated science ınquiry ıntegrated with mind map. *Journal of Turkish Science Education*, 14(4):77–91, 2017.

Drill core image recognition with three-dimensional attention and self-calibration

Yong Pu[1], Chuanhu Xiong[1], Yonghua Chen[1], Ziyuan Xu[2]
[1]Guangzhou Metro Design & Research Institute Co., Ltd., Guangzhou, China
[2]Institute of Intelligence Science and Technology, School of Computer Science and Software Engineering, Hohai University, Nanjing, China
{puyong, xiongchuanhu, chenyonghua}@dtsjy.com, hideon27@hhu.edu.cn

Abstract

Drill core image recognition is an extremely critical aspect of geological exploration. However, the research on drill core image recognition faces challenges such as subtle discrepancies between different lithological categories, complex backgrounds, unclear recognition subjects, and lack of datasets. Currently, the existing recognition methods for geological images struggle to accurately classify drill core images. To address these issues, this paper constructs a drill core image dataset called Drill Core Dataset (DCD), and proposes an image recognition model called MSL-ResNet50, which combines self-calibration residual module and three-dimensional attention. By introducing a self-calibration module with a larger receptive field, the model captures higher-level semantic information and enhances the network's feature transformation capability. Furthermore, it integrates the SimAM attention mechanism to mitigate interference in context information, giving higher attention to key areas. Additionally, the MAM pooling strategy is utilized to alleviate the loss of image features and maintain consistency in feature distribution of feature maps before and after pooling. Extensive ablation and validation experiments were conducted on the DCD dataset. The experimental results demonstrate that the proposed model, MS-ResNet50, significantly outperforms the related benchmark models in drill core image recognition. Specifically, it achieves a recognition accuracy of 99.3%, an improvement of 4.1% over the original model, validating the effectiveness of the proposed model for core image recognition tasks.

Index terms— ***Drill core image recognition, self-calibration, three-dimensional attention, MAM pooling***

1 Introduction

Image recognition is a key technology in computer vision to automatically identify and analyze objects and features in digital images. Drill core image recognition is a process specifically applied to geological exploration and petroleum engineering, aiming to automatically analyze core images to extract geological information. Traditional methods in drill core recognition rely on the experiences and technical capabilities of field surveyors, which is time-consuming, labor-intensive and highly subjective. Furthermore, differences in professional proficiency would lead to uneven accuracy of drill core identification results. Nowadays, automatic core image recognition can be achieved through deep learning.

Early lithology identification utilized the principal component analysis (PCA) to reduce the dimensionality of the original correlated lithology parameters [1, 2, 3]. Recently, deep learning has been applied to strata identification. Cheng proposed a particle size analysis method for rock images based on convolutional neural networks [4]. Zhang et al. accomplished the identification of three types of rocks by building a deep convolutional neural network model based on Inception-v3 [5]. Liu et al. proposed an optimal method for automatic identification of thin argillaceous interlayers, providing reference for the efficient development of oil sand projects [6]. Alférez et al. employed the convolutional neural network (CNN) model developed by TensorFlow to classify granite rocks [7]. Xu et al. proposed the Faster R-CNN architecture for rock image prediction, based on the ResNet structure and retaining the detailed information of the original image through residual learning [8]. Chen et al. established a classification framework for tunnel face rock structure based on the Inception-ResNet-V2 convolutional neural network [9]. Zhang et al. presented an improved Branch Module structure based on the AlexNet network to promote recognition accuracy and reduce the number of model parameters [10]. Bharadiya et al. combined a CNN model for image classification with learning representation to tackle the shortcomings of traditional feature selection methods [11]. Baraboshkin et al. applied CNN to the field of rock classification and suggested that GoogLeNet and ResNet are architectures with

DOI reference number: 10.18293/DMSVIVA24-008

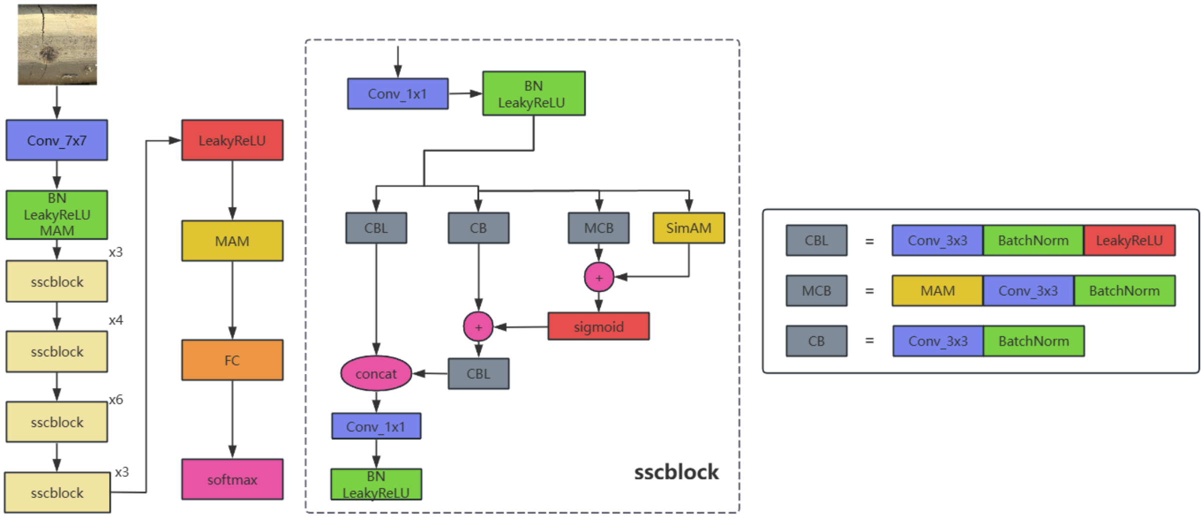


Figure 1: The structure of the proposed model MS-ResNet50.

preferable performance [12]. Zhang et al. developed an intelligent system for identifying continental shale lithology using data balancing and deep learning, with the EfficientNet model performing best and providing technical support for evaluating continental shale reservoirs [13]. Wang et al. constructed an improved lightweight MobileViT model to analyze rock slice images covering most common lithology, addressing issues of unbalanced lithology datasets and numerous identification model parameters [14].

Despite remarkable progress in lithology recognition research, some challenges remain in this field. First, the images sampled at different times and locations would exhibit significant variations due to environmental influences, which places high demands on the generalization ability of the model. Second, since many rocks are highly similar in appearance and composition, the lithology of rocks presents the issue of large intra-class differences and small inter-class differences, making accurate identification difficult. Third, in dealing with diverse samples acquired from complex geological conditions, existing methods have difficulty in accurately separating and identifying each lithology in strata with mixed lithologies. This could be especially true when the lithology data has a highly unbalanced distribution.

To address the above challenges, on the basis of ResNet [13], a deep network that has been proven by the above-mentioned existing work to be effective in tackling lithology identification, this paper proposes an improved deep network model MS-ResNet50 dedicated to the identification of drill core images, by optimizing the residual structure, pooling strategy and activation function of the original model. The main contributions of the paper are summarized as follows:

1. A dataset for drill core images, called Drill Core Dataset (DCD), is constructed. It includes 3070 drill core images, which are sampled under various environments, weather conditions, shooting angles, lighting, and depths.

2. A residual network that combines SimAM and self-calibrated convolution is constructed. The latent space is utilized to calibrate the original space to obtain more surrounding context information, including the contour and texture of the strata. The three-dimensional attention mechanism SimAM is employed to assist the latent space to extract information in a targeted manner, thus eliminating the interference in context information.

3. The MAM pooling strategy is introduced. The mean of each pooling window is compared with the standard deviation and mean of the entire pooling domain to select the appropriate pooling method, so that the feature map after pooling maintains the same feature distribution as the original image, and meanwhile obtains subtle features and significant features, and reduces the information loss caused by the pooling operation, thus improving the ability to distinguish between similar drill cores.

4. Extensive experiments on the data set DCD are conducted. The experimental results demonstrate that the proposed model significantly outperforms the existing models in lithology recognition. In particular, the accuracy of the improved model reaches 99.3%, which is 4.1% higher than the original model.

The remainder of the paper is organized as follows. Section 2 provides the overall architecture of the proposed model, Section 3 presents the self-calibrating convolution module with three-dimensional attention mechanism, Section 4 introduces the MAM pooling method, Section 5 conducts experiments and result analysis, and finally Section 6 concludes the paper.

2 MODEL ARCHITECTURE

The architecture of the proposed model MS-ResNet50 is shown in Figure 1. There are three fundamental components appearing in the network: CBL, MCB and CB, where CBL stands for Convolution, Batch normalization and LeakyReLU, MCB stands for MAM pooling, Convolution and Batch normalization, and CB stands for Convolution and Batch normalization. It incorporates three crucial improvements on the original model ResNet50.

As an improvement to the conventional residual module, MS-ResNet50 constructs a module called sscblock that combines the self-calibrated convolution and the SimAM attention mechanism. This module divides the input into two parts. One part is the same as that of ResNet50, performing a standard 3×3 convolution, and the other is divided into two paths. Path 1 obtains the context information of the surrounding area after downsampling as a latent space with large receptive field, whereas Path 2 extracts features in the original scale space. Then, the outputs from the two paths are added and convolved to obtain the calibrated feature map. Since the self-calibration strategy would inevitably bring in useless background information, the SimAM mechanism is introduced in the latent space to acquire the context information, and the cross-dimensional information obtained from the interaction between the three dimensions of space and channel can be leveraged to reduce the interference of background information.

To tackle the issue that the original pooling method may cause information loss when reducing the model size, an improved strategy is proposed. Specifically, average pooling is not conducive to the extraction of edge features, and maximum pooling merely focuses on obvious features and ignores details and texture features, resulting in serious feature loss. Therefore, the MAM pooling method is employed to replace the original pooling, setting different pooling strategies in terms of the pooling window and the global situation. The flexible selection of pooling strategies can keep the style of the feature map before and after pooling consistent, and meanwhile enhances the model's perception of image details.

Additionally, the original activation function ReLU is replaced by LeakyReLU, to alleviate the issue of neuron "necrosis". A comparative experiment has validated that LeakyReLU is the most beneficial activation function for enhancing the recognition performance of the model.

3 Self-calibrated convolution module with 3D attention mechanism

Traditional convolution operations are performed by sliding a fixed-size convolution kernel over the input feature map. For a set of convolution kernels $K = [k_1, k_2, \ldots, k_c]$, and input data $X = [x_1, x_2, \ldots, x_c\ ]$, the output data after convolution is denoted as $Y = [y_1, y_2, \ldots, y_c]$. The output of the i-th channel can be expressed as $y_i = k_i * X = \sum_{j=1} k_i^j * x_j$.

This kind of convolution uses the same formula to calculate the output of each channel, and the final feature map is obtained by summing these outputs. Consequently, the features learned by the convolution kernels often lacks diversity, and the extracted feature map also lacks distinctiveness. In addition, the predefined convolution kernel size determines the receptive field of each spatial position, making it challenging for the network to effectively capture the high-level semantic information and generate optimal output results. The network constructed using such convolution layers exhibit clear drawbacks, such as insufficient receptive field and inadequate grasp of high-level semantic information.

Given that the feature transformation capability of a network is one of the critical factors affecting the recognition performance of convolutional neural networks, this section introduces a feature extraction method based on a self-calibrated residual network. This method leverages an internal message-passing mechanism to break away from the traditional use of small-size kernels for information fusion and extraction in both channel and spatial dimensions without adding additional learnable parameters.

However, when using self-calibrated convolutions to acquire surrounding context information, irrelevant background information would inevitably interfere with the network. Therefore, to assist the network in focusing attention on key beneficial areas, a self-calibrated convolution module with a three-dimensional attention mechanism is proposed. The module captures the critical information overlooked across three dimensions, avoiding the interference caused by long-range context information, and helps the model better understand the overall structure and context of the image.

3.1 Self-calibrated residual module

The structure of the Self-Calibrated Convolution (SCConv) is shown in Figure 2. It divides the convolution kernels of a specific layer into multiple parts, which are then fed into two different scale spaces for feature transformation. This allows for more effective capture of the rich context information surrounding each spatial position. In SCConv module, the self-calibration operation can be utilized to adaptively adjust the learning of long-distance spatial positions and the interaction between channels, thereby achieving the objective of expanding the receptive field of convolutions. The heterogeneous convolution communication means significantly expands the receptive field of each spatial position, thereby improving the perception ability of the network.

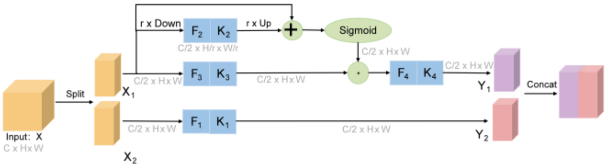


Figure 2: The structure of self-calibrated convolution module.

The specific calculation steps are as follows:

(1) The size of the input feature map is $C \times H \times W$. It is split into two parts, each with a size of $C/2 \times H \times W$, and sent into different paths to collect different types of context information.

(2) There are 4 convolution kernels, denoted as K_1, K_2, K_3, and K_4, each with dimensions $C/2 \times H \times W$.

(3) Processing the self-calibrated scale space:

For X_1, there are two parts: one entering the original space and the other entering the latent space. In the latent space, feature X_1 undergoes average pooling with a 4x downsampling:

$$T_1 = AvgPool_r\left(X_1\right) \tag{1}$$

Then, bilinear interpolation is used for upsampling, mapping the small-scale space back to the original feature space:

$$X_1' = Up(F_2(T_1)) = Up(T_1 * K_2) \tag{2}$$

where $*$ indicates the convolution operation. Then, a residual structure is constructed by addition and passed through the sigmoid activation function:

$$M_1 = \sigma\left(X_1 + X_1'\right) \tag{3}$$

where σ is the sigmoid function, and $X_1^{'}$ serves as the residual to form the calibration weight. In the original space, X_1 is passed into the K_3 convolution kernel and calibrated using the features obtained from the latent space to produce Y_1:

$$\begin{aligned} Y_1' &= F_3\left(X_1\right) \bullet M_1 = \left(X_1 * K_3\right) \bullet M_1 \\ Y_1 &= F_4\left(Y_1'\right) = K_4 * Y_1' \end{aligned} \tag{4}$$

where $F_3(X_1) = X_1 * K_3$, and $\bullet$ represents element-wise multiplication. Y_1 is the final output after calibration.

(4) Processing the original space: perform a K_1 convolution operation on feature X_2 to extract feature Y_2;

(5) Concatenate the output features Y_1 and Y_2 from the two scale spaces to obtain the final output feature Y.

3.2 Three-dimensional attention mechanism

The essence of the attention mechanism lies in the process of to dynamically select information in the input image using different weights. In the current research on the combination of attention mechanisms and neural networks, CBAM (Convolutional Block Attention Module) is a representative attention mechanism module, which sequentially connects the channel attention module and the spatial attention module. Its structure is shown in Figure 3.

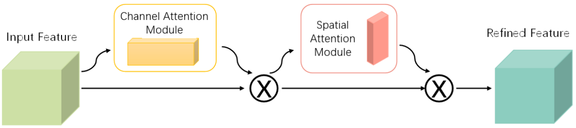


Figure 3: The CBAM structure.

The CBAM module can serially generate attention feature map information in the channel and space dimensions for the input feature map, and then achieve adaptive feature correction by multiplying it with the original input feature map. The structures of the channel attention and spatial attention are shown in Figure 4 and Figure 5, respectively.

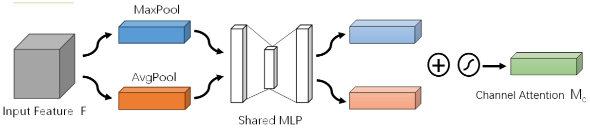


Figure 4: The channel attention structure.

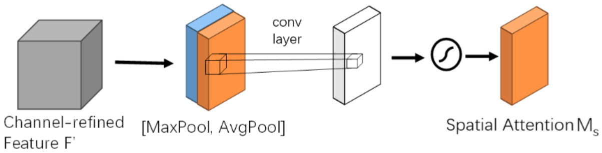


Figure 5: The spatial attention structure.

In terms of neuroscience theory, human spatial attention and channel attention often coexist and interact, jointly promoting visual processing. However, according to the above analysis, although the CBAM module takes into account the feature information of both spatial and channel dimensions, this serial approach makes the channel and spatial channels relatively independent, neglecting the interaction between them, which leads to the loss of important cross-dimensional information.

The SimAM mechanism can make up for the above shortcomings of the CBAM module. SimAM assigns a unique weight to each neuron without adding additional parameters, and treats the channel and spatial attention operations equally. This strategy generates three-dimensional weights that amplify the interaction features across all dimensions and reduce information diffusion.

Neuroscientific research indicates that if a neuron carries a large amount of information, its discharge pattern tends to significantly differ from that of other surrounding neurons. When these neurons are activated, they usually cause inhibition of surrounding neurons, a property known as spatial inhibition. In other words, in a neural network, neurons with spatial inhibition property should be given more attention and assigned higher weights compared to other neurons. The simplest way to identify such neurons is to measure the linear separability between neurons.

According to neuroscience theory, an energy function can be defined for each neuron to evaluate its importance, as shown in Equation 5:

$$e_t\left(w_t, b_t, y, x_i\right)=\left(y_t-\hat{t}\right)^2+\frac{1}{M-1} \sum_{i=1}^{M-1}\left(y_o-\hat{x_i}\right)^2 \tag{5}$$

where M represents the number of neurons in each channel, with $M = H \times W$; t denotes the target neuron and x_i denotes other neurons on the same channel as t; $\hat{t}$ indicates a linear transformation of t defined as $\hat{t} = w_t + b_t$, and $\hat{x_i}$ represents a linear transformation of x_i defined as $\hat{x_i} = w_t x_i + b_t$, where w_t and b_t are the weights and biases assigned during the linear change; and y_o and y_t represent binary labels.

In order to train the linear separability between the target neuron and other neurons in the same channel, binary labels are used, with y_t=1 and y_o=-1, and a regularization term is added to minimize Equation 5, yielding the final energy function:

$$\begin{aligned} e_t\left(w_t, b_t, y, x_i\right)=\frac{1}{M-1} \sum_{i=1}^{M-1} & \Big[\left(-1-\left(w_t x_i+b_t\right)\right)^2 \\ & +\left(1-\left(w_t+b_t\right)\right)^2\Big]+\lambda w_t^2 \end{aligned} \tag{6}$$

Theoretically, each channel has M neurons, resulting in M energy functions per channel. To reduce the computational overhead, the analytical solutions of w_t and b_t are derived:

$$\begin{aligned} w_t &= -\frac{2\left(t-\mu_t\right)}{\left(t-\mu_t\right)^2+2 \sigma_t^2+2 \lambda} \\ b_t &= -\frac{1}{2}\left(t+\mu_t\right) w_t \\ \mu_t &= \frac{1}{M-1} \sum_{i=1}^{M-1} x_i \\ \sigma_t^2 &= \frac{1}{M-1} \sum_{i=1}^{M-1}\left(x_i-\mu_t\right)^2 \end{aligned} \tag{7}$$

where μ_t represents the mean of all other neurons in the channel except for the target neuron, and σ represents the variance of all other neurons in the channel except for the target neuron. Since all neurons in each channel follow the same distribution, the minimum energy function for each position can be expressed as:

$$e_t^*=\frac{4\left(\hat{\sigma}^2+\lambda\right)}{(t-\hat{\mu})^2+2 \hat{\sigma}^2+2 \lambda} \tag{8}$$

Equation 8 indicates that the linear separability between the target neuron and other neurons in the same channel is inversely proportional to its energy value. That is, when a neuron bears a lower energy, it is more distinctive from other neurons, and the importance of the neuron will be greater.

After determining the importance of neurons, the SimAM attention mechanism is used to enhance the features, resulting in a new feature map $\widetilde{X}$, as shown in Equation 9. Here, E represents the set of all energy values in the input feature map, and "$\odot$" is the dot product. The Sigmoid function is applied to limit the impact of excessively large values in E.

$$\widetilde{X}=sigmoid\left(\frac{1}{E}\right) \odot X \tag{9}$$

Building on the improved self-calibrated residual network discussed in Section 3.1, this section incorporates the non-parametric SimAM as the attention module within the network. This method not only enhances the fusion of the spatial and channel information without incurring additional computational costs, but also reduces the introduction of unimportant feature information when integrating surrounding context information through self-calibration. The structure of the improved self-calibration residual module with the SimAM attention is depicted in Figure 6.

4 MAM Pooling

The primary objective of pooling is to reduce the number of parameters of the model, minimize the interference

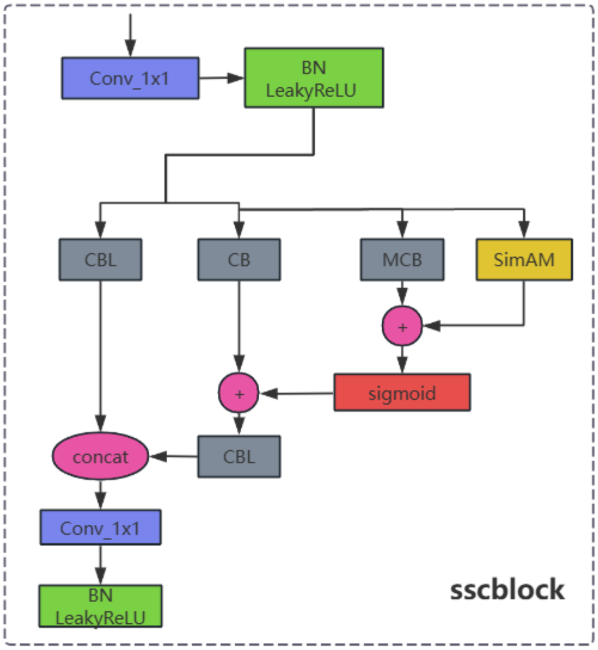


Figure 6: The structure of self-calibration residual with the SimAM attention.

of redundant information, and simultaneously retain critical feature information as much as possible, thereby maintaining the invariance of the feature map before and after pooling [13]. Currently, commonly used pooling methods include maximum pooling and average pooling. However, these two methods result in feature maps to be either close to its mean or close to its standard deviation after pooling, making it difficult to balance both.

When pooling feature maps, the approach that favors retaining the prominent features or overall style of the image leads to the incomplete understanding of the image. Retaining a part of the minimum values during pooling can preserve subtle features in the image and reduce the loss of overall features, thereby keeping the style and content features of the image before and after pooling unchanged. Additionally, the feature map after pooling shows clear contrasts, allowing for the exaction of sharp edges.

Therefore, it is necessary to design a dynamically selectable pooling method. To ensure that the feature map maintains consistent in content features and style before and after pooling, the mean $Avg(a_{ij})$ of each pooling window is compared with the weighted mean m_A and standard deviation s_A of the entire pooling region to determine whether to retain the maximum value, the average value, or the minimum value for each pooling window. The specific pooling process is expressed as follows:

$$B_{ij} = \begin{cases} Max(a_{ij}) & Avg(a_{ij}) > m_A + \alpha s_A, \\ Avg(a_{ij}) & m_A + \alpha s_A \geq Avg(a_{ij}) \geq m_A - \beta s_A \\ Min(a_{ij}) & Avg(a_{ij}) < m_A - \beta s_A. \end{cases} \tag{10}$$

Here i and j represent the i-th row and j-th column of the feature map after pooling, A and B indicates the feature maps before and after pooling, respectively, and are adjustable parameters, and the pooling window size and step size are both k.

When $Avg(a_{ij})$ is greater than the weighted sum of m_A and s_A, it means that the data in the pooling window tends to be larger values. In this case, selecting the maximum pooling can better retain the prominent features. When $Avg(a_{ij})$ is less than the weighted difference between m_A and s_A, it means that the data in the pooling window tends to be smaller values. In this case, selecting the minimum pooling can better retain the subtle features. When $Avg(a_{ij})$ is between the weighted difference and the weighted sum of m_A and s_A , selecting the average pooling can better integrate the feature information within the window, thereby extracting a more comprehensive representation.

This strategy, at the cost of introducing a small amount of additional computation, can effectively choose the appropriate pooling method in terms of the overall situation of the data in the pooling window.

5 Experimental results and analysis

5.1 Datasets

We collected images of drill cores from different geological environments, including images taken at different time periods after drilling and from different sections, to comprehensively reflect the characteristics of cores, and constructed the Drill Core Dataset (DCD) accordingly. By retaining the influence of various factors such as different humidity levels, angles, and background interference, the dataset is made to be closely aligned with real-world application scenarios. The DCD dataset contains 15 representative rock and soil sub-layer categories, such as silt, silty soil, fine sand, medium-coarse sand, gravel sand, etc., totaling 3,070 images.

5.1.1 Dataset Analysis

(1) Uneven data distribution

Figure 7 lists the number of samples for each category in the DCD dataset. As shown in the figure, the average number of samples per category is 264. The category with the most samples is plastic clay (4N-2), having 796 samples, while the category with the fewest samples is silty soil (2-1B), having 172 samples. Noticeably, the number of samples in the plastic clay (4N-2) category is significantly higher than in other categories. We applied data augmentation to all sample images and balanced the categories with large discrepancies in sample numbers.

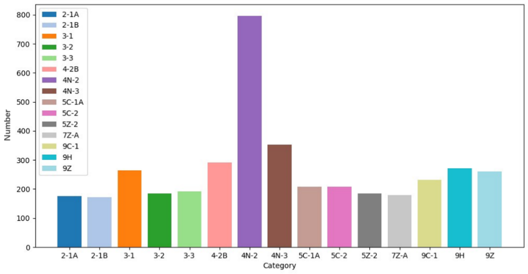


Figure 7: Number of samples in each category.

(2) Large intra-class differences, small inter-class differences

Through observation and comparison, it has been found that for the same type of drill core, different periods of photography exhibit varying humidity, texture, and shape characteristics. The same category of drill cores often includes two or more forms, with significant differences between them. Figure 8 illustrates silt clay with diverse colors and textures.

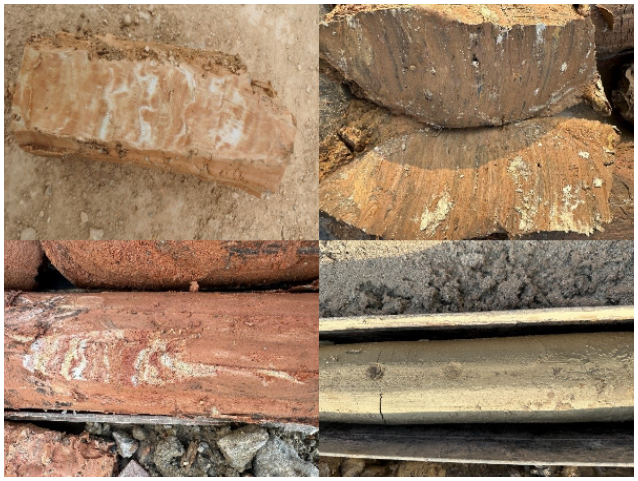
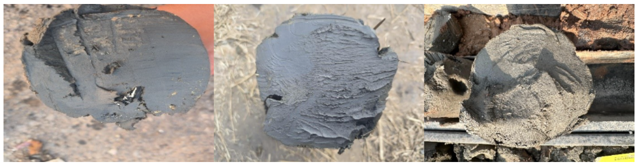


Figure 8: Examples of silty clay drill cores with diverse colors and textures.

For drill core images of different sublayers, there are often extremely similar forms that are difficult to distinguish. Figure 9 shows images of three categories: silt (2-1A), silty soil (4-2B), and silty clay (4N-2). Obviously, they have similar colors, identical shapes, and subtle inter-class differences.



(a) silt (b) silty soil (c) silt clay

Figure 9: Examples of drill cores for different categories.

5.2 Experimental setup

The experimental settings and parameters are as follows: Adam is used as the optimizer, cross entropy loss function is utilized, batch size is 32, learning rate is 0.0001, number of epochs is 200, and the ratio of training set to validation set is 9:1. The preprocessing operations on the images before training include random horizontal flipping, random cropping, adding Gaussian noise, etc.

5.3 Experimental results and analysis

In this section, the results of extensive experiments are discussed from four perspectives. The first part is the set of comparative experiments of residual networks with different depths, the second part is the set of comparative experiments for different activation functions, the third part is the ablation study of the improved modules, and the fourth part is the set of comparative experiments of the proposed model and the baseline models.

(1) Comparative experiments of residual networks with different depths

The set of experiments was conducted using residual networks with different numbers of layers, and the experimental results are shown in Table 1.

Table 1. Experimental results of residual networks with different depths.

Model	Top-1 Accuracy (%)
ResNet34	93.9
ResNet50	**95.2**
ResNet101	94.7

From Table 1, it can be seen that ResNet50 performs best on the DCD dataset. In addition, the 101-layer ResNet performs even worse than the 50-layer ResNet. This phenomenon is likely due to the small size of the DCD dataset, where a deeper network structure may lead to overfitting, resulting in performance degradation.

(2) Comparative experiments of activation functions

In this set of experiments, the activation function in the ResNet50 model were replaced with PReLU, ELU, GELU, and LeakyReLU, respectively, for comparison. The experimental results are shown in Table 2.

Table 2. Residual networks with different activation functions.

Activation Function	Top-1 Accuracy (%)
ReLU	95.2
PReLU	95.6
ELU	94.3
GELU	94.0
LeakyReLU	**96.4**

The results show that the activation functions LeakyReLU and PReLU perform well on the ResNet50 model, enhancing the model's recognition performance to some extent. By examining the computational efficiency and cost of both, LeakyReLU was chosen as the activation function due to its higher efficiency.

(3) Ablation study

In order to verify the effectiveness of each improved module, ablation experiments were conducted on the DCD dataset, including three models. Model 1 is a ResNet50 network using LeakyReLU as the activation function (L-ResNet50), Model 2 is formed by adding SCConv and SimAM modules to model 1 (SL- ResNet50), and Model 3 is formed by using MAM pooling on the basis of model 2 (MSL- ResNet50). The experimental results of the three models on the DCD dataset are shown in Table 3. Adding both the self-calibrated convolution SCConv and the three-dimensional attention mechanism SimAM, and the MAM pooling mechanism to the ResNet50 model with improved loss function, results in varying degrees of improvement in the network's recognition accuracy, with increases of 1.9% and 1.0% respectively.

Table 3. Ablation study on the DCD dataset.

Number	Model	Top-1 Accuracy (%)
1	L-ResNet50	96.4%
2	SL-ResNet50	98.3%
3	MSL-ResNet50	**99.3%**

Each of the three models was trained for 200 epochs, and the running results of validation accuracy of these models on the DCD dataset are shown in Figure 10.

Table 4 shows the recognition results of the MSL-ResNet50 model on all the categories.

(4) Comparative experiments of related models

To further validate the classification performance of the proposed model for drill core images, we compared the proposed model MSL-ResNet with several representative benchmark models. The benchmark models include some mainstream image classification deep network models, such as AlexNet [15], GoogleNet [16], Vgg16 [17], Vision-Transformer [18], and the classification model ConvNet for plutonic rocks. In the experiments, all models used the same dataset partitioning, training epochs, and initial learning rate. Table 5 exhibits the classification results of the involved models and Figure 11 depicts the curves of validation accuracy of them. It can be observed that the MSL-ResNet50 proposed in this paper achieves significantly superior classification accuracy.

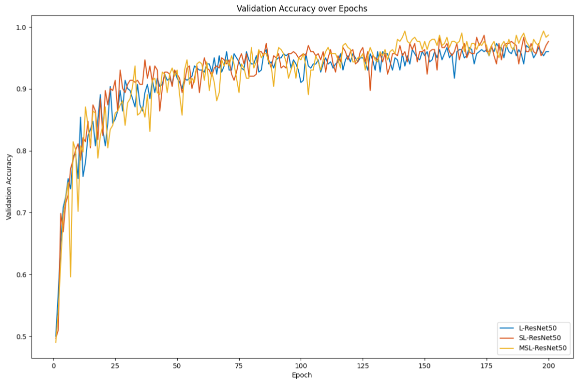


Figure 10: The curves of validation accuracy of the three ResNet50 models.

Table 4. The recognition precision and recall of MSL-ResNet50 for each category.

Category	Precision	Recall
2-1A	0.941	0.941
2-1B	1.0	0.941
3-1	1.0	1.0
3-2	0.9	1.0
3-3	1.0	0.947
4-2B	0.954	0.954
4N-2	1.0	1.0
4N-3	1.0	1.0
5C-1A	1.0	1.0
5C-2	1.0	1.0
5Z-2	1.0	1.0
7Z-A	1.0	1.0
9C-1	1.0	1.0
9H	1.0	1.0
9Z	1.0	1.0

6 Conclusion

This paper has proposed an improved model, MSL-ResNet50, for drill core image recognition. On the basis of ResNet50, a self-calibrated residual structure is first introduced, which leverages a latent space with a larger receptive field to calibrate the original space so as to obtain more context information. Then, a simple parameter-free

Table 5. Experimental results of involved models for comparison.

Number	Model	Top-1 Accuracy (%)
1	AlexNet	93.4
2	GoogleNet	86.6
3	VGG16	87.3
4	Vision-Transformer	68.3
5	ConvNet	74.3
6	MSL-ResNet50	99.3

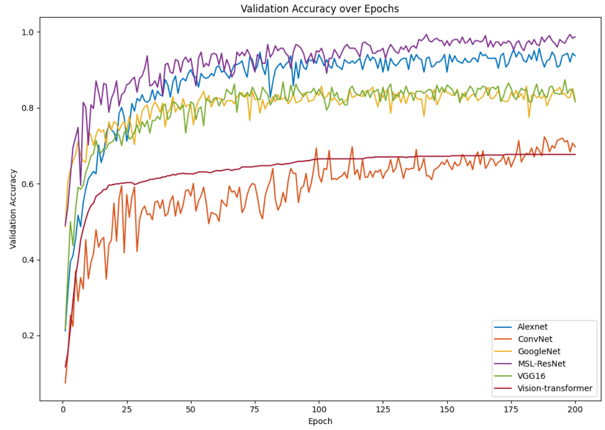


Figure 11: The curves of validation accuracy of the involved models for comparison.

attention mechanism, SimAM, is integrated into the model to further optimize the self-calibrated residual structure, thereby avoiding the interference from the acquired context information. Third, the MAM pooling method, which has demonstrated preferable performance, is applied within the model. The MAM method can dynamically select the pooling strategies for the network, preserving important detailed features and alleviating the issue of feature loss. Additionally, a drill core dataset DCD was established. Extensive experiments were conducted on the DCD dataset, and the results manifested that the proposed model, MS-ResNet50, significantly outperforms the related benchmark models in drill core image recognition. Specifically, the proposed model achieves a recognition accuracy of 99.3%, indicating an improvement of 4.1% over the original model, which validates the effectiveness of the improved model in the task of drill core image recognition.

References

[1] Y. Zhong and R. Li. Lithology identification method based on principal component analysis and least squares support vector machine. *Well Logging Technology*, 33(5):5, 2009.

[2] Y. Zhang and B. Pan. Application of principal component analysis-based som neural network in volcanic rock lithology identification. *Well Logging Technology*, 33(6):550–554, 2009.

[3] A. Liu, L. Zuo, J. Li, et al. Application of principal component analysis in carbonate rock lithology identification — a case study of cambrian carbonate reservoir in YH area. *Petroleum and Natural Gas Geology*, 34(2):192–196, 2013.

[4] G. Cheng and W. Guo. Rock images classification by using deep convolution neural network. In *Journal of Physics: Conference Series*, volume 887, page 012089. IOP Publishing, 2017.

[5] Y. Zhang, M. Li, and S. Han. Lithology automatic recognition and classification method based on rock image deep learning. *Acta Petrologica Sinica*, 34(2):333–342, 2018.

[6] Y. Liu, J. Huang, Y. Yin, et al. Optimization and application of core image recognition algorithms for oil sand reservoirs. *Fault-Block Oil & Gas Field*, 27(4):464–468, 2020.

[7] G. H. Alférez, E. L. Vázquez, A. M. M. Ardila, et al. Automatic classification of plutonic rocks with deep learning. *Applied Computing and Geosciences*, 10:100061, 2021.

[8] Z. Xu, W. Ma, P. Lin, et al. Deep learning of rock images for intelligent lithology identification. *Computers & Geosciences*, 154:104799, 2021.

[9] J. Chen, T. Yang, D. Zhang, et al. Deep learning based classification of rock structure of tunnel face. *Geoscience Frontiers*, 12(1):395–404, 2021.

[10] B. Zhang. *Research on neural network-based core image recognition algorithms*. PhD thesis, Yangtze University, 2022.

[11] J. Bharadiya. Convolutional neural networks for image classification. *International Journal of Innovative Science and Research Technology*, 8(5):673–677, 2023.

[12] E. E. Baraboshkin, L. S. Ismailova, D. M. Orlov, et al. Deep convolutions for in-depth automated rock typing. *Computers & Geosciences*, 135:104330, 2020.

[13] Z. Zhang, J. Tang, B. Fan, et al. An intelligent lithology recognition system for continental shale by using digital coring images and convolutional neural networks. *Geoenergy Science and Engineering*, 239:212909, 2024.

[14] Q. Wang, J. Yang, F. C. Huo, et al. Lithology identification method of rock thin section images based on mobilevit. *Geological Bulletin of China*, 43(6):938–946, 2024.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[16] C. Szegedy, W. Liu, Y. Jia, et al. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

API Knowledge Graph Construction Based on Multi-Source Information Fusion

Ruilian Zhao[1], Zijie Che[1], Zhan Ma[1], Weiwei Wang[2]
[1]Beijing University of Chemical Technology, Beijing, China
[2]Beijing Institute of Petrochemical Technology, Beijing, China
Email: rlzhao@mail.buct.edu.cn, wangweiwei@bipt.edu.cn

***Abstract*—API-related knowledge is typically dispersed across various sources of information, including API documentation, Q&A forums, and other unstructured texts. This fragmentation of knowledge makes it challenging for developers to effectively query and retrieve APIs. In this paper, an API knowledge graph construction method based on multi-source information fusion is proposed to overcome these issues and enhance API retrieval. Specifically, the API-related knowledge is acquired from multiple sources, including API documentation and Stack Overflow, where API documentation describes the function and structure of APIs from designers' perspective, and Stack Overflow provides insights into the purpose and usage scenarios of APIs from users' perspective. They complement each other and together provide support for API query and retrieval. By analyzing API documentation, the corresponding APIs and domain concepts are extracted as entities and relationships between them are identified. Moreover, to extract Q&A entities from Stack Overflow, machine learning is adopted to classify the purpose of the question and performs the summary generation for its answers. Since there exists a gap between the entities from API documentation and Stack Overflow, a fusion method is raised to establish connections between them, constructing a more comprehensive API knowledge graph. To verify the effectiveness of our API knowledge graph construction method, we evaluate it in terms of the accuracy of knowledge extraction and API recommendation. The experimental results demonstrate that our API knowledge graph can significantly improve the efficiency and effectiveness of API recommendation.**

***Index Terms*—API knowledge graph, Multi-source information, Knowledge extraction, Knowledge fusion, API retrieval**

I. Introduction

API (Application Programming Interface) plays a critical role in software development. According to statistics, 87% of developers frequently leverage APIs to address diverse programming issues [1]. However, retrieving and finding suitable APIs is still a challenging task. To improve the efficiency and quality of API retrieval, researchers have built corresponding API recommendation systems from various resources to assist developers in solving programming issues related to APIs.

Currently, several typical API recommendation systems have been developed, such as RASH [2], BIKER [1], RACK [3] etc. RASH leverages lexical similarity to recommend APIs based on API documentation and the Stack Overflow (*SO*). In contrast, BIKER utilizes semantic similarity by combining API documentation and *SO* to recommend APIs. RACK establishes relationships between keywords in titles of *SO* and API to recommend APIs. These methods have enhanced retrieval efficiency in contrast to conventional API retrieval. However, they solely concentrate on valid APIs utilized in resolved problems and overlook the interconnection between APIs. In fact, different types of relationships between APIs, such as inheritance between classes and invocation between methods, may have varying impacts on API recommendation. Additionally, APIs that resolve identical problems may possess functionally similar relationships with one another, which could enhance the effectiveness of API retrieval. Nevertheless, the previous API recommendation techniques have not fully leveraged such relationships.

The Knowledge Graph is a knowledge network that can effectively represent the semantic association between information, which is suitable for expressing API-related knowledge. For example, Liu et al. [4] constructed an API knowledge graph by extracting relevant knowledge from API documentation and Wikipedia, while Li et al. [5] constructed an API warning knowledge graph by extracting warning statements from API documentation and API tutorials. Ling et al. [6] constructed an API knowledge graph based on open-source projects, which took APIs involved in projects as entities, and the calls, returns and implementations between APIs as relationships. As can be seen, API documentation can provide the dependency between APIs, Wikipedia can provide the concept of software engineering, and open-source projects can provide the relationship (call, return, implementation etc.) between APIs. However, all of them lack descriptions on actual API usage scenarios, which hinders the practical use of API knowledge graphs in solving real programming issues.

Stack Overflow is an IT technical Q&A website for programmers. It aims to help solve the actual problems of developers, and provide information about the purpose of API usage and real usage scenarios [7]. If the actual usage scenarios of APIs in *SO* are incorporated into the knowledge graph, it will greatly facilitate API retrieval for users. But *SO* suffers from a lack of clarity of purpose and information overload. The statistics show that more than 37% of *SO* questions contain more than one answer, with an average of more than 789 words per answer [8]. This makes it more difficult to capture useful knowledge from *SO*.

Thus, this paper proposes an API Knowledge Graph construction based on Multi-Source Information Fusion (AKG-MSIF), which synthesizes APIs and usage scenarios from

DOI reference number: 10.18293/DMSVIVA2024-027

API documentation and stack overflow. In particular, it entails extracting API and domain concepts as entities from API documentation and establishing relationships, such as inclusion, inheritance, and overloading, between them. More importantly, for *SO*, to extract its Q&A entities, our method uses machine learning to classify the purpose of the question and performs the summary generation for its answers. On this basis, multi-source knowledge is integrated to construct an API knowledge graph. Since there exists a gap between the entities from API documentation and *SO*, a fusion method is raised to establish connections between them. To validate the effectiveness and efficiency of our method, the constructed API knowledge graph is evaluated from two perspectives: knowledge extraction accuracy and recommendation effect. The experimental results show that compared with existing studies, our AKG-MSIF approach improve the API recommendation effectiveness and efficiency. Our contributions are as follows:

1. A novel API knowledge graph is constructed by integrating information from both API documentation and *SO*, facilitating API retrieval for users.
2. Due to unclear purpose and information overload in *SO*, a machine learning-based method is raised to classify the purpose of the question and performs the answer summary generation to obtain the Q&A entities. In addition, a knowledge fusion methods are raised to bridge the gap between entities of API documentation and *SO*.
3. To validate our approach, a series of experiments are conducted, and the experimental results show that compared with existing API recommendation systems, our novel knowledge graph has enhanced the recommendation effectiveness and efficiency.

The rest of this paper is organized as follows: Section 2 introduces the background of related techniques. Section 3 describes our method in detail. Section 4 verifies the validity of the approach. Section 5 summarizes the whole paper.

II. Related Work

Currently, several typical API recommendation systems have been developed, such as RASH [2], BIKER [1], RACK [3] etc. RASH leverages lexical similarity to recommend APIs based on API documentation and the Stack Overflow. In contrast, BIKER utilizes semantic similarity by combining API documentation and Stack Overflow to recommend APIs. RACK establishes relationships between keywords in titles of Stack Overflow and API to recommend APIs. These methods have enhanced retrieval efficiency in contrast to conventional API retrieval.

Ye et al. [9] proposed a rule-based entity extraction method, which mostly uses keywords, central words, superlatives, subordinate words, punctuation marks and other features in the text. This approach relies on the creation of a complete knowledge base and lexicon. Stephen et al. [10] proposed an approach for entity extraction through NLP and pattern-matching to classify Stack Overflow sentences. This approach also requires the design of extraction rules. Unlike the rule-based approach, it takes the syntax and grammar of the text as the focus, converts it into a syntactic dependency tree through NLP techniques and analyzes its dependencies, thereby obtaining the structural parts of the text such as noun phrases and verb phrases. For example, Lin et al. [11] manually defined 157 grammatical templates for the language style of the Stack Overflow. It can be seen that this method works better for texts with more uniform content formatting.

III. API Knowledge Graph Construction Based on Multi-Source Information Fusion

In this paper, we propose an API knowledge graph construction based on multi-source information fusion, where the API-related knowledge derives from API documentation and *SO*. The framework of our approach is shown in Fig.1, which mainly consists of knowledge acquisition and knowledge fusion. Concretely, in knowledge acquisition, APIs and corresponding domain concepts are extracted from the API documentation and taken as entities. And relationships between them, such as inclusion, inheritance, and overloading, are established. Furthermore, Q&A and API concepts are identified from *SO* by using machine learning and regarded as entities. And relationships between Q&A and API concepts are built. In knowledge fusion, multi-source knowledge from API documentation and *SO* is integrated to construct a more comprehensive API knowledge graph based on these entities and relationships. Since there exists a gap between the entities from API documentation and *SO*, the relationship between them is established by various fusion strategies. In the following, we will detail each part of our approach.

A. Knowledge Acquisition from API Documentation

API documentation provides functional descriptions and structural information (such as method, parameters and return values etc.) for APIs. This part focuses on the knowledge representation and extraction of API documentation about the PyTorch framework.

1) Knowledge representation of API documentation: The structural information of the API refers to modules, classes, methods/functions, etc., which are related to each other by inclusion, inheritance, overloading, etc. Moreover, the functional description in API documentation implies the application domains of the API, which indirectly reflect the relationship between the API and application domain. Both the functional description and structural information can provide useful guidance in API retrieval. Thus, we use the domain concept and API modules, classes, methods/functions to express the API knowledge in the document. Further, the APIs and domain concepts can be associated through ”refer to”, which means that the description of an API mentions the corresponding domain concept. So, this paper regards the API and domain concepts as entities in the API documentation and their ”refer to” as the relationship between them.

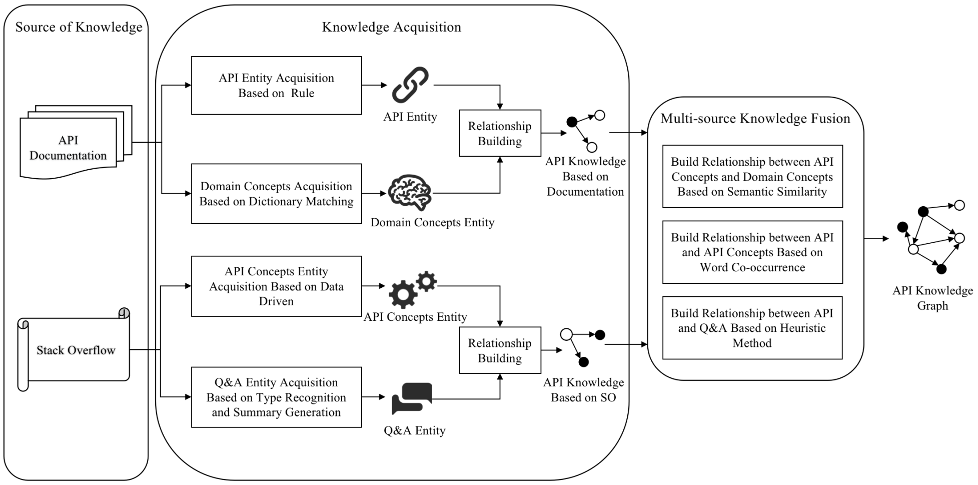

Source of Knowledge
API Documentation
Stack Overflow
Knowledge Acquisition
API Entity Acquisition Based on Rule
API Entity
Domain Concepts Acquisition Based on Dictionary Matching
Domain Concepts Entity
Relationship Building
API Knowledge Based on Documentation
API Concepts Entity Acquisition Based on Data Driven
API Concepts Entity
Q&A Entity Acquisition Based on Type Recognition and Summary Generation
Q&A Entity
Relationship Building
API Knowledge Based on SO
Multi-source Knowledge Fusion
Build Relationship between API Concepts and Domain Concepts Based on Semantic Similarity
Build Relationship between API and API Concepts Based on Word Co-occurrence
Build Relationship between API and Q&A Based on Heuristic Method
API Knowledge Graph

Fig. 1. The Framework of API Knowledge Graph Based on Multi-Source Information Fusion

2) Knowledge extraction from API documentation: To extract API entities, the API documentation is analyzed. And we found that API documentation is semi-structured data, where different HTML tags represent different types of API entities, such as functional descriptions, parameters, return values, and return value types etc. Thus, API entities are recognized through HTML tags. Further, the relationships between API entities include inclusion, inheritance, overloading. According to the declaration rules of the class, regular expressions are used to extract the inheritance relationship, and syntax analysis is employed to extract the inclusion and overloading relationship.

Furthermore, each API corresponds to a functional description, and the functional description implies domain concepts. Thus, for the application domain in API function descriptions, we use existing domain concept dictionary [12] and NLP to match and recognize them. And the ”refer to” relationship between the API and the domain concept can be extracted from this corresponding structure. For example, the functional description of API “*torch.normal*” contains the domain concept “*standard deviation*”. When the domain concept “s*tandard deviation*” is identified, a ”refer to” relationship can be established between the API ”*torch.normal*” and the domain concept ”*standard deviation*”.

B. Knowledge Acquisition from Stack Overflow

Stack Overflow provides many information (such as title, the body of the question, label, accepted answer etc.) which contains specific application scenario information of the API. This part mainly focuses on the knowledge representation and extraction on the Q&A tagged by ”PyTorch” on the Stack Overflow.

1) Knowledge Representation of Stack Overflow: The Q&A information of *SO* contain terms related to the API, where these terms are related to software development, without being limited to a specific field. Intuitively, these terms implicitly abstract and summarize the functional role of a specific API. Thus, these terms are adopted to express the API-related knowledge and regarded as API concept entities.

What’s more important, the Q&A in *SO* describe the actual problems encountered by developers and provide answers about its usage scenario as well as the purpose of API. So the Q&A is critical in API retrieval. But the Q&A in *SO* suffers from problems of unclear purposes and information overload. The unclear purpose refers to the difficulty for Q&A to grasp the reason behind a user’s question and find the corresponding answer. And information overload refers to the fact that a single question may have multiple long answers. Statistics show that over 37% of Q&As include more than one answer, and each answer has an average of over 789 words [8], making it challenging to obtain critical information from the Q&A.

Thus, in this paper, we propose a questions’ purpose identification method through classifying the Q&A automatically. Further, the answers are summarised based on the features of Q&A to alleviate the issue of information overload. And the simplified Q&As that consists of the purpose of questions and summary of answers are referred to as Q&A entities. Besides, the “refer to” relationship between Q&A entities and API concept entities can be established.

2) Knowledge extraction from Stack Overflow: The API in *SO* is usually labeled with <*code*> tags. Thus, API can be recognized through matching the element labeled with this tag with the API name in the API documentation. For the concepts of API in *SO*, they often appear in the same sentence, paragraph, or Q&A with the API. Since the API concept may be a multi-word concept, such as ”convolutional layer”, this paper proposes a frequency-based API concept recognition method. Concretely, we look for words that often appear consecutively but not often separately in *SO* through NLP, and take them as API concepts. NLP is used to segment and remove stop words from *SO* Q&A to form single-word concepts. According to the frequency of consecutive words, we calculate the phrase score and extract the API concept.

The phrase score is shown in formula (1):

$$score(w_i w_j) = \frac{count(w_i w_j) - \delta}{count(w_i) \times count(w_j)} \quad (1)$$

Where $count(w_i w_j)$ represent the number of times two consecutive words w_i and w_j appear in the whole documentation. $count(w_i)$ and $count(w_j)$ represent the number of times the words w_i and w_j appear. δ is a threshold. When the frequency of the two consecutive words w_i and w_j is less than δ, w_i and w_j cannot form a two-word phrase. When a two-word concept is formed, formula (1) can be repeated to detect three-word phrases. Since API concepts consisting of more than three words are uncommon, this paper only recognizes phrases of up to three words as API concepts.

To extract the Q&A entities from *SO*, this paper employs machine learning to classify the purpose of the questions and obtain the purpose type. Based on this, the answer summary generation based on feature extraction is performed.

In more detail, based on the categorization of *SO* Q&A by Stefanie Beyer et al.'s [13], this paper divides the Q&A into seven categories based on the purpose of question as follows: (1) "API USAGE" class is to seek suggestions for implementing a feature or API; (2) "DISCREPANCY" class is to request Code segments to resolve unexpected results; (3) "ERRORS" class is to request a bug fix or handle an exception; (4) "REVIEW" class is to request the best solution; (5) "CONCEPTUAL" class is to ask about the rationale or background of the API; (6) "API CHANGE" class is to seek solutions to issues arising from API version changes problems; (7) "LEARNING" class is to ask for documentation or tutorials to learn a tool or language.

XGBoost(eXtreme Gradient Boosting) is one of machine learning algorithms, which have the capability of fast learning and prediction [14]. Therefore, in this paper, XGBoost algorithm is used to train classifiers for *SO* questions to determine the purpose of questions. The main steps include: (1) Label *SO* Q&A into one of the seven categories. (2) Convert questions into corresponding word lists through NLP including as segmentation, stop word removal, and lemmatization. (3) The TF-IDF reflects the importance of a word by its frequency, where *TF* (term frequency) measures the frequency that a term appears in a document and *IDF* (the inverse document frequency) estimates the ratio of total documents to the documents that contain the term. In this paper, the TF-IDF of a question is used as its textual feature and fed into the XGBoost algorithm to identify the type of the question.

Furthermore, to address information overload in answers of *SO*, this paper generates summaries for answers based on the relevant paragraphs in the answers. That is, based on the characteristics of *SO*, the relevance of each paragraph to the question is calculated by combining question-related features, content-related features, and user features, and the top M paragraphs are selected as the summary of the answer. The concrete feature analysis is as follows:

(1) Question-related feature: if a paragraph contains key words from the question, it is considered to be related to the question. The more key words a paragraph contains, the higher its relevance. In this paper, tags of *SO* are used as the set of key words. And the relevance of each answer paragraph and the question is calculated based on the ratio of the key words involved in them.

(2) Content-related feature: This feature evaluates the importance of content of paragraphs from three sub-features: the API occurrence, information entropy, and semantic templates. For the API occurrence, if at least one API appears in the paragraph, this sub-feature value is set to 1. Otherwise, it is set to 0. For information entropy, the inverse documentation frequency (IDF) value of a word can be used to measure its information entropy, which can be calculated using formula (2), where p represents the total number of paragraphs and p' represents the number of paragraphs containing a particular word. The higher the IDF value, the lower the occurrence frequency of the particular word, indicating greater importance. The entropy value of a paragraph can be represented by the sum of its words' IDF values, normalized to (0,1].

If a paragraph conforms to at least one semantic template, the sub-feature value is set to 1. Otherwise, it is set to 0. The feature value of the content is the sum of the three sub-feature values.

$$IDF = log(\frac{p}{p' + 1}) \quad (2)$$

(3) User feature: In *SO*, each answer has a corresponding vote, and the higher the vote, the higher the quality of the answer. Therefore, the number of votes for the current answer indicates the importance of the paragraph in this answer, which can be regarded as the user feature.

For the above three features, we add a smoothing factor of 0.0001 to avoid the feature score of 0. All features are normalized to (0,1], and the normalized values of each feature are multiplied together to obtain the total score of each paragraph. Finally, the top M paragraphs are selected as the summary of the answer.

By identifying the type of question and generating the answer summary, an valid Q&A entity can be obtained. Furthermore, since a Q&A usually mentions multiple API concepts, a "refer to" relationship can be also established between the API concept and the Q&A entities.

C. Knowledge Fusion from API Documentation and Stack Overflow

To construct a complete API knowledge graph, the API knowledge from API documentation and *SO* Q&A website should be integrated. As there is a gap between entities from the API documentation and *SO*, a fusion method is proposed to establish a link between them. As mentioned above, entities about APIs and corresponding domain concepts are extracted from the API documentation. And entities about API concepts and Q&A are extracted from *SO*. Since domain concepts are not directly related to Q&A entities, it is not mandatory to establish a connection between them. Thus, this paper performs knowledge fusion between entities about API and

API concept, API concept and domain concept, and API and Q&A.

1) Fusion between entities of API and API concepts based on word co-occurrence: Intuitively, API concepts abstract and summarize the functional role of a specific API. Thus, semantic relationships exist between them. In fact, API and API concepts usually co-occurs in the same paragraph, so word co-occurrence can be used to link them. Co-occurrence frequency can evaluate the degree of correlation between API and API concepts, which refers to the number of times the API and API concepts appear in the same paragraph. Therefore, this paper captures the semantic relationship between API and API concepts by calculating their co-occurrence frequency. Its formula is shown in formula (3), where $freq(A_i \rightarrow Ac_j)$ represents the co-occurrence frequency between API A_i and API concept AC_j, and α is the threshold. If the co-occurrence frequency is not lower than the threshold α, a "refer to" relationship can be established between API A_i and API concept AC_j.

$$freq(A_i \rightarrow AC_j) \geq \alpha \tag{3}$$

2) Fusion between entities of API concept and domain concept based on semantic similarity: The relationship between API concepts and domain concepts can help establish indirect connections between the API, which can help improve the possibility of retrieving relevant APIs. Thus, it is necessary to build the links between them. Since API concepts and domain concepts are composed of phrases, their relationship can be determined by combining lexical and semantic similarity. When the similarity between them is higher than the given threshold, their "related to" relationship can be established.

In more detail, the lexical similarity sim_{lex} can be calculated using Jaccard similarity, as shown in formula (4), where $Token(n)$ represents the words that make up the concept. The semantic similarity between n_1 and n_2 is calculated using formula (5), where $V_p(n_1)$ represents the vector of the concept entity, and sim_{cos} represents the cosine similarity between the two vectors. In this paper, based on SO Q&A and API documentation corpora, we use word2vec [15] to train a word embedding model and convert concepts into word vectors. Based on the lexical and semantic similarity, a weighted similarity calculation formula is raised, which is shown in formula (6). Generally, semantic similarity is more important than lexical similarity, so $w_1 < w_2$ is set.

$$sim_{lex}(n_1, n_2) = \frac{|Token(n_1) \bigcap Token(n_2)|}{|Token(n_1) \bigcup Token(n_2)|} \tag{4}$$

$$sim_{con}(n_1, n_2) = \frac{sim_{cos}(V_p(n_1), V_p(n_2)) + 1}{2} \tag{5}$$

$$sim(n_1, n_2) = w_1 \times sim_{lex}(n_1, n_2) + w_2 \times sim_{con}(n_1, n_2) \tag{6}$$

Formula (4) measures the similarity between domain concepts and API concepts in terms of both lexical and semantic aspects, where n_1 and n_2 represent the candidate domain concept and API concept, respectively.

3) Fusion between entities about API and Q&A based on heuristic algorithm: The relationship between API entity and Q&A entity enables the integration of the general knowledge of the API (such as functional description, parameters, return values, etc.) with their specific knowledge in concrete usage scenarios (such as how to solve specific problems), which provides developers with a more comprehensive API information. Thus, a fusion strategy is raised to establish the relationship between them.

However, APIs mentioned in *SO* Q&A are not always in the form of fully qualified names. For example, the API "forward()" is mentioned in *SO* in answer to the question "what does model.train() do in PyTorch". But "forward()" can be associated with multiple APIs. In order to establish an unambiguous correlation between Q&A entities and corresponding API entities, we design a heuristic strategy. In general, in *SO* Q&A, the appearance of code elements has locality, i.e., APIs mentioned in the same Q&A usually belong to the same module or class. By parsing the tag in HTML, we can identify the module or class of the API mentioned in the Q&A. By specifying regular expressions to identify the module or class in the code block, their APIs can be determined. Once unambiguous APIs are identified, a "refer to" relationship can be established between the Q&A entity and the API entity.

IV. Experimental Analysis

In order to verify the validity of our AKG-MSIF approach for API retrieval, we conduct a series of experiments on PyTorch API documentation and 7043 API Q&A on Stack Overflow, and the effectiveness and efficiency are evaluated on the basis of these experiments. To assess our approach, three research questions are raised as below.

- **RQ1.** Can the API knowledge be accurately extracted from multi-source information?
- **RQ2.** Can our integrated API knowledge graph obtained by fusing API documentation and *SO* improve the effectiveness of API retrievals?
- **RQ3.** How effective is our AKG-MSIF approach in the API recommendation? How much improvement can be achieved compared to baseline methods?

A. Experimental Subject

In this paper, we extracted questions and answers marked as "PyTorch" from the official *SO* data (data released as of June 2022). We extracted 7043 questions and answers labeled as "PyTorch" as the subjects. In order to ensure the quality of the Q&A, we excluded the Q&A with no answer and those with a rating of less than 1 (indicating that the content of the answer was not accepted), and finally collected 3361 Q&A with good quality. Besides, we develop a crawler script based on scrapy framework, and obtain information about the API by crawling the official API documentation of PyTorch. In total, 27 modules, 314 classes, 1570 functions or methods and their corresponding basic description information are extracted. The fully qualified names of these API classes and functions/methods are used to build the API dictionary of PyTorch.

The final constructed API knowledge graph includes 28730 entities and 142,578 relations. Among them, there are 1912 API entities, 16216 API concepts, 7116 domain concepts, 3361 Q&A entities.

B. Experimental Design

When extracting API concepts, the threshold δ of the frequency of the two consecutive words was set to 5 to avoid the recognition of uncommon phrases. Furthermore, when integrating API and API concepts, the co-occurrence frequency threshold is set to 3 to capture the semantic association between entities about API and API concept. And when integrating API concepts and domain concepts, considering that semantic similarity is more important than lexical similarity, weights w_1 and w_2 were set to 0.3 and 0.5 respectively.

Besides, in order to create experimental queries for retrieving knowledge graph, the following selection criteria were used: 1) The questions had a rating at least 1. 2) The answer to the question contains the explicit and exact API and the title of the question does not contain the API. Based on them, 10 questions were randomly selected from the PyTorch-related questions in *SO* as the queries for the experiment, and the corpus for constructing the knowledge graph did not contain these 10 questions in order to ensure that the search of API knowledge graph was valid.

The API knowledge graph constructed in this paper is stored in a Neo4j graph database. For queries, corresponding keywords are extracted by syntactic analysis using the StanfordCoreNLP[21]. And for each keyword, we search a semantically similar concept entity in the API knowledge graph. The API entity with a "refer to" relationship with the concept entity is used as a candidate API, and the Q&A associated with the candidate API is information about the specific usage scenario. Since there may be multiple candidate APIs, they are ranked according to their semantic relevance to the query. That is, the APIs are ranked by calculating the semantic similarity (formula(5)) between the query and each candidate API function description, and the top K APIs are recommended to users.

The related APIs obtained by searching the API knowledge graph are further analyzed, so as to verify the effectiveness of API recommendation based on our knowledge graph. In particular, we invite 10 masters from the same lab with two years of experience in using PyTorch to analyze the accepted or highly rated responses to these questions together with the authors themselves. When disagreements arose, consistent conclusions were drawn by analyzing the official API documentation. The final 10 experimental queries and the number of correct APIs for them are shown in Table 2.

C. Experimental Results and Analysis

1) Results for RQ1: The focus of this experiment is to demonstrate the effectiveness of knowledge extraction from the perspective of entities and relationships extraction. Thus, the accuracy of entities and relationships extracted from the API documentation and *SO* is evaluated.

As is known, API entities and their relationships are derived from semi-structured API documentation. Based on specific HTML tags and declarations, entities and relationships related to them can be extracted and validated easily. Therefore, this paper mainly evaluates the extraction accuracy of entities from unstructured text, namely API concept, domain concept and Q&A entities, as well as their relationships. Since the number of domain concept entities and relations exceeds tens of thousands, it takes a lot of time to check all entities and relations. Thus, this paper adopts the random sampling. In more detail, random samples of 5% of the entities or relationships from the constructed API knowledge graph is selected with a 95% confidence level, and the sample estimation accuracy has an error margin of 0.05.

To assess the validity of API concepts and domain concepts, we manually identifying the accuracy of sampling results. After random sampling, the accuracy of 356 domain concepts obtained from domain concept entities is up to 95.6%, and the error mainly comes from the domain concept dictionary itself. The accuracy of 800 API concepts sampled from API concept entities is 97.8%, and the main reasons affecting the accuracy are some numerical indicators often mentioned in *SO* Q&A, such as "200k images". These terms should not be identified as API concepts. To evaluate the validity of the relationship between the API concept and domain concept, the accuracy of sampling results is also manually identified. After random sampling, 4000 relationships were obtained from the API knowledge graph, of which 94.3% of API concepts and domain concept semantics were identified as relevant. The missing relationships are due to API concepts or domain concepts not being correctly identified.

To evaluate the effectiveness of the Q&A entity extraction, the accuracy of the classification of *SO* questions and the quality of answer summary is measured. For the classification of *SO* question, the XGBoost algorithm is used to classify the question types. Through manual labeling, 326 labeled Q&A were obtained, including 118 "API USAGE", accounting for 36.2%, and 65 "CONCEPTUAL", accounting for 20%; 45 "DISCREPANCY", accounting for 13.9%; 34 "ERRORS", accounting for 10.4%; 24 "REVIEW", accounting for 6.1%. The number of "API CHANGE" and "LEARNING" is 20, accounting for 6.1%. In this paper, a 10-fold cross-validation method was used to verify the validity of *SO* Q&A classification. The classification effectiveness was evaluated using precision, recall, F1 value and accuracy. To verify the advantages of XGBoost-based classification, the experiment uses SVM (Support Vector Machine) and RF (Random Forest) as comparison methods. The comparison of classification effectiveness of different algorithms are shown in Table II. The precision of XGBoost is improved by 14.6% and 5.7% compared to SVM and RF, respectively, and the accuracy is improved by 5.9% and 4.6%, respectively. Therefore, it can be seen that the classification for questions of *SO* using XGBoost

TABLE I
QUERIES AND THE NUMBER OF STANDARD ANSWERS

SO Number	Question	Number of Related API
44524901	How to do product of matrices in PyTorch?	6
54716377	How to do gradient clipping in PyTorch?	2
48152674	How to check if PyTorch is using the GPU?	7
50544730	How do I split a custom dataset into training and test datasets	1
55546873	How do I flatten a tensor in PyTorch?	4
53841509	How does adaptive pooling in PyTorch work?	4
53266350	How to tell PyTorch to not use the GPU?	2
53879727	PyTorch-How to deactivate dropout in evaluation mode?	2
51136581	How to do fully connected batch norm in PyTorch?	4

algorithm is better than other methods.

TABLE II
COMPARISON OF CLASSIFICATION EFFECTIVENESS OF DIFFERENT ALGORITHMS

Method	Precision	Recall	F1 Score	Accuracy
XGBoost	0.871	0.847	0.834	0.910
SVM	0.760	0.851	0.785	0.850
RF	0.824	0.903	0.849	0.870

Furthermore, to evaluate the quality of summary generation, this part measures the quality of summary in terms of relevance, usefulness, and diversity. Relevance indicates whether the summary is relevant to the question. Usefulness indicates whether the summary content can solve the problem, and diversity indicates whether the question can be answered from multiple perspectives. In this part, we set a maximum score of 5 and a minimum score of 1 for each indicator. To evaluate the quality of the summary, master students with two years of experience using PyTorch were invited to participate in the evaluation. Table III represents the evaluators' assessment of the 10 *SO* Q&A summaries in Table I, where the three evaluation metrics for each query ranged from 3 to 5, and the average results for the 10 questions were 3.6, 3.4, and 3.7, respectively, indicating that the feature extraction-based summary generation method is effective.

2) Results for RQ2: To validate whether the integrated API knowledge graph obtained by fusing API documentation and *SO* improve the effectiveness of API retrievals, we compare the retrieval results based on multi-source API knowledge graph with single-source API knowledge graph. The single-source API knowledge graph is constructed by extracting API-related knowledge from API documentation and *SO* Q&A websites, respectively. The single-source API knowledge graph extracted from the API documentation includes API entities, domain concept entities and the relationships among them; And the other single-source API knowledge graph extracted from *SO* Q&A website includes API entities, API concept entities, Q&A entities and the relationships among them. Three commonly used metrics in information retrieval are selected to evaluate the effectiveness of API retrieval, namely, HR(Hit Ratio), MRR(Mean reciprocal rank) and MAP(Mean average precision). *HR* evaluates the percentage of correct results out of all correct results in the top K search results. *MRR* is the position where the first correct result appears. *MAP* is the ranking of all correct results. Since the number of the API related to query is less than 10, this paper sets K=10.

The experimental results are shown in Table IV. It can be seen that the API recommendation effectiveness of multi-source API knowledge graph is better than that of single-source API knowledge graph. This is because that information fusion indirectly associates the API related to the questions through the mentioned concepts, which improves the recommendation effectiveness. In addition, it is worth noting that there is a large difference between the recommendation results based on API documentation and those based on *SO* Q&A websites. The reason is that the functional descriptions provided by the API documentation do not involve specific usage scenarios. Thus, it is difficult to match the domain concepts with the keywords in the specific questions, resulting in unsatisfactory recommendation results by using only the API documentation. In a summary, we can see that our API knowledge graph is more comprehensive, and enhances the effectiveness of API retrieval, indicating our AKG-MSIF approach is effective.

3) Results for RQ3: To evaluate the effectiveness of our AKG-MSIF approach in API recommendation, three metrics including HR, MRR and MAP are also used. Besides, the BIKER recommendation system also combines two types of data sources (API documentation and Stack Overflow), and it uses the same dataset as the approach in our paper. While RACK recommendation system uses only Stack Overflow data sources. These two techniques are used as our comparison methods. The experimental results are shown in Table V. It can be seen that the HR index of our AKG-MSIF has increased by 49% compared with BIKER and 87% compared with RACK. The MRR index has increased by 22% compared with BIKER and 52% compared with RACK. This indicates that in the first 10 search results, AKG-MSIF can search more APIs related to the query, and can find the first correct API earlier than BIKER and RACK. Thus, our AKG-MSIF has improved retrieval efficiency compared with BIKER and RACK. Besides, Table VI shows the comparison in terms of time cost. The construction time of AKG-MSIF is mainly concentrated in the training of the classifier and summary generation. Although the time cost of the construction of the method in this paper is higher than that of BIKER and RACK, there is a significant improvement in the query speed and the recommendation effectiveness of the API. Thus, our AKG-MSIF approach is more effective in API retrieval.

TABLE III
Q&A SUMMARY SCORE

Question	Relevance	Usefulness	Variety
How to do product of matrices in PyTorch?	3	3	3
How to do gradient clipping in PyTorch?	2.85	3	3.57
How to check if PyTorch is using the GPU?	4	3.5	3.5
How do I split a custom dataset into training and test datasets	3.5	4	3.5
How do I flatten a tensor in PyTorch?	3.57	2.85	3.57
How does adaptive pooling in PyTorch work?	3.5	3	4
How to tell PyTorch to not use the GPU?	4.5	4	4.5
PyTorch-How to deactivate dropout in evaluation mode?	3	3	3.5
How to do fully connected batch norm in PyTorch?	4	3.5	4
How to create a normal distribution in PyTorch?	4.5	4	4
AVERAGE	**3.6**	**3.4**	**3.7**

TABLE IV
COMPARISON OF RECOMMENDATION RESULTS BETWEEN MULTI-SOURCE INFORMATION FUSION AND SINGLE-SOURCE INFORMATION

Method	HR	MAP	MRR
Only SO	0.726	0.490	0.583
Only API Doc	0.322	0.181	0.149
Both	0.774	0.558	0.701

TABLE V
COMPARISON OF RECOMMENDATION EFFECTIVENESS BY DIFFERENT METHODS

Method	HR	MAP	MRR
AKG-MSIF	0.774	0.558	0.701
BIKER	0.520	0.521	0.573
RACK	0.415	0.420	0.462

TABLE VI
COMPARISON OF TIME COST BY DIFFERENT METHODS

Method	Cost	Query Cost
AKG-MSIF	16 min	1s/query
BIKER	5 min	2s/query
RACK	10min	5s/query

V. CONCLUSION

This paper proposes an API knowledge graph construction approach based on multi-source information fusion(AKG-MSIF), which integrates the functional and structural information of APIs, as well as the specific usage scenarios of APIs from documentation and *SO* Q&A websites. The experiment validated the effectiveness of our approach from two aspects: information extraction and API recommendation effectiveness. And the results show that the accuracy of domain concept identification is up to 95.6%, and that of API concepts is 97.8%. And 94.3% of relationships between API concepts and domain concept are correctly identified. Meanwhile, the Q&A entities from *SO* are identified effectively by machine learning and summary generation. Furthermore, compared with existing API recommendation systems, our API knowledge graph is more comprehensive, enhancing the effectiveness of API retrieval.

REFERENCES

[1] Q. Huang, X. Xia, Z. Xing, D. Lo, and X. Wang, "Api method recommendation without worrying about the task-api knowledge gap," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 293–304.

[2] J. Zhang, H. Jiang, Z. Ren, and X. Chen, "Recommending apis for api related questions in stack overflow," *IEEE Access*, vol. 6, pp. 6205–6219, 2017.

[3] M. M. Rahman, C. K. Roy, and D. Lo, "Rack: Automatic api recommendation using crowdsourced knowledge," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1. IEEE, 2016, pp. 349–359.

[4] M. Liu, X. Peng, A. Marcus, Z. Xing, W. Xie, S. Xing, and Y. Liu, "Generating query-specific class api summaries," in *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 120–130.

[5] H. Li, S. Li, J. Sun, Z. Xing, X. Peng, M. Liu, and X. Zhao, "Improving api caveats accessibility by mining api caveats knowledge graph," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2018, pp. 183–193.

[6] C.-Y. Ling, Y.-Z. Zou, Z.-Q. Lin, and B. Xie, "Graph embedding based api graph search and recommendation," *Journal of Computer Science and Technology*, vol. 34, pp. 993–1006, 2019.

[7] C. Treude and M. P. Robillard, "Augmenting api documentation with insights from stack overflow," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 392–403.

[8] S. Nadi and C. Treude, "Essential sentences for navigating stack overflow answers," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 229–239.

[9] D. Ye, Z. Xing, J. Li, and N. Kapre, "Software-specific part-of-speech tagging: An experimental study on stack overflow," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016, pp. 1378–1385.

[10] S. Soderland, "Learning information extraction rules for semi-structured and free text," *Machine learning*, vol. 34, pp. 233–272, 1999.

[11] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, and M. Lanza, "Pattern-based mining of opinions in q&a websites," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 548–559.

[12] C. Wang, X. Peng, M. Liu, Z. Xing, X. Bai, B. Xie, and T. Wang, "A learning-based approach for automatic construction of domain glossary from source code and documentation," in *Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 97–108.

[13] S. Beyer, C. Macho, M. Pinzger, and M. Di Penta, "Automatically classifying posts into question categories on stack overflow," in *Proceedings of the 26th Conference on Program Comprehension*, 2018, pp. 211–221.

[14] S.-E. Ryu, D.-H. Shin, and K. Chung, "Prediction model of dementia risk based on xgboost using derived variable extraction and hyper parameter optimization," *IEEE Access*, vol. 8, pp. 177 708–177 720, 2020.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

MVKGCL: Recommendation Model Based on Knowledge Graph and Contrastive Learning

Zhiyue Xiong[1], Hankiz Yilahun[1*], Sadiyagul Anwer[2], Askar Hamdulla[1]

(1. School of Computer Science and Technology, Xinjiang University, Urumqi 830017, China;

2. Xinjiang Agricultural Vocational Technical University, Urumqi 830017, China)

***Abstract*—Most existing recommendation models based on knowledge graphs and contrastive learning employ random augmentation schemes to enhance the data in knowledge graphs; however, noise in knowledge graphs can lead to inaccurate recommendation results. Furthermore, most contrastive learning methods are only applied between one or two views, thereby failing to exploit the semantic information in the data fully. Therefore, this model proposes a recommendation model, MVKGCL, which integrates knowledge graphs and contrastive learning mechanisms. Firstly, it incorporates random noise into attention weights to conduct contrastive learning among different attention weights, and subsequently introduces a novel automatic masking mechanism to augment the Knowledge Graphs, performing local contrastive learning on the derived user and item embeddings. Secondly, it employs Graph Attention Network to encode the user-item-entity graph, yielding representations for users and items. Lastly, global level contrastive learning is conducted between the locally learned user and item embeddings and the node embeddings from the user-item-entity graph, uncovering comprehensive graph features and structural information. Experiments demonstrate that this model outperforms others on the Amazon-book and Yelp2018 datasets, with average increases of 3.5% and 4.4% in Recall, and 2.4% and 2.6% in NDCG, respectively.**

Keywords—Knowledge graphs; Recommendation model; Contrastive learning; Graph attention network.

I. Introduction

As networks have rapidly advanced and contemporary tech products have become widespread, humanity has entered the epoch of big data, giving rise to colossal volumes of data in everyday life. The capability of users to handle information lags significantly behind the pace at which information disseminates, a dilemma termed the issue of information overload. Recommendation models automatically assist users in pinpointing pertinent information amidst this sea of data, furnishing them with tailored data services. Fundamentally, collaborative filtering recommendation algorithms hinge on scrutinizing user conduct, item characteristics, and the historical interplay between users and items [1,2]. By doing so, they distil the traits of users and items, facilitating individualized recommendations tailored to diverse users. Even though these algorithms endeavor to model intricate dynamics between users and items, numerous models grounded in collaborative filtering grapple with sparse data complications and the cold start predicament. Knowledge graphs (KG), brimming with substantial entities and relational insights, can potentially augment the semantic depictions of both users and items. Consequently, incorporating KG as supplementary data into recommendation frameworks serves as a remedy for these hurdles and bolsters overall efficacy [3]. Nonetheless, KG is beset by issues about noise and a shortage of dense supervisory cues [4].

Inspired by contrastive learning's approach of mining supervisory signals from the data itself, this model focuses on exploring a multi-view contrastive learning mechanism to alleviate the challenges above. The main contributions of this model are as follows (shown in Figure 1):

- Incorporates random noise into attention weights to conduct contrastive learning among different attention weights, and introduction of a novel automatic masking mechanism for data augmentation of KG, enhancing the consistency of KG-augmented subgraphs to fortify user-item interaction graphs.
- Implementation of local-level contrastive learning between KG and user-item graphs, fostering a detailed contrastive understanding at the granular level.
- Employment of graph attention mechanisms for high-order semantic encoding of user-item-entity graphs, assigning varying weights to nodes, thereby generating embeddings for users and items that reflect their unique roles and relationships.
- Conduction of global-level contrastive learning between locally embedded nodes and the node embeddings within the user-item-entity graph, yielding more nuanced node embeddings. This process enriches representations and mitigates issues of data sparsity and noise prevalent in recommendation models.

II. Related Work

A. Knowledge-aware Recommendation

The embedding based method [5] uses Knowledge Graph Embedding (KGE) [6,7] to preprocess KG, and then incorporates the learned entity embeddings and relationship embeddings into recommendations. Collaborative Knowledge Base Embedding (CKE) [8] combines the CF module with project structure, text, and visual knowledge embedding in a unified Bayesian framework. KTUP [9] considers the incompleteness of knowledge graphs when using them for recommendation algorithms, and combines learning recommendation and knowledge graph completion. This method proposes a TUP (translation based user preference)

*Corresponding author: Hankiz Yilahun (hansumuruh@xju.edu.cn)

DOI reference number: 10.18293/DMSVIVA2024-050

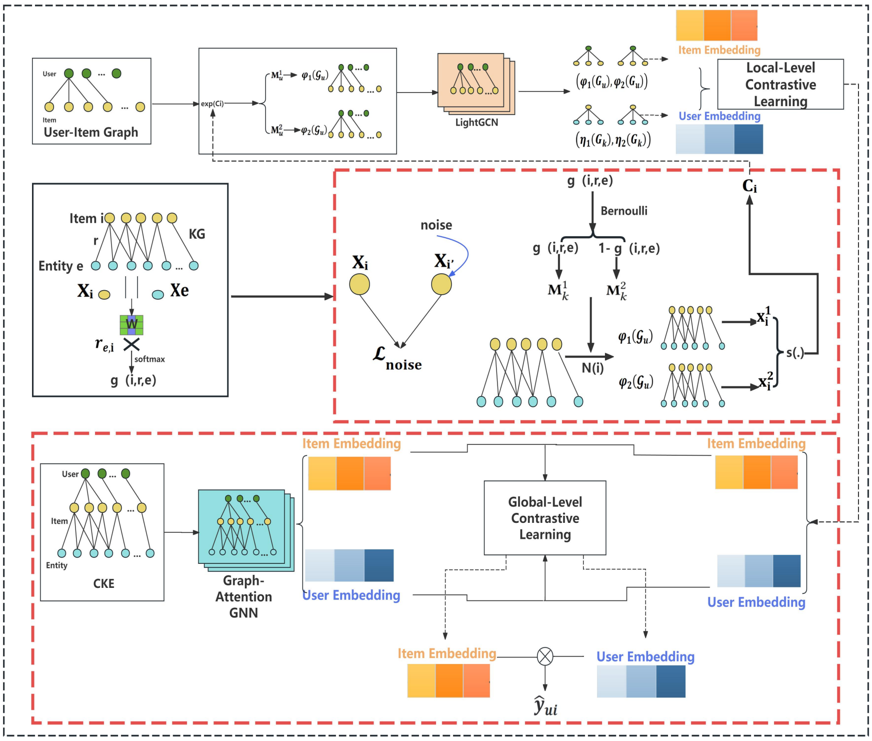


Fig. 1. MVKGCL model architecture

model combined with knowledge graph learning, and utilizes multiple implicit relationships between users and items to reveal user preferences. KTUP combines TUP and TransH [10] for joint learning, enhancing project and preference modeling by transferring entity knowledge and relationships. RippleNet [11] is a classic recommendation algorithm based on knowledge graph propagation mechanism. In RippleNet, the items that users interact with are called seeds, and each seed propagates in the knowledge graph, spreading to other entities, thereby extending and expanding user interests. The embedding based method demonstrates high flexibility in utilizing KG, but the KGE algorithm focuses more on modeling strict semantic correlations (e.g. TransE [12] assumes head+relation=tail),which is more suitable for link prediction rather than recommendation.

The method of graph-based information aggregation mechanism neural networks (GNNs) [13,14] integrates multi hop neighbors into node representations to capture node features and graph structures, thus simulating long-range connectivity. KGCN [15] combines knowledge graph and graph convolutional neural network to effectively capture local neighborhood information and consider neighbor node weights for recommendation. This model samples the neighboring nodes of candidate items in the knowledge graph, and then iteratively samples the neighboring nodes for each entity, using a linear combination of neighboring node information to characterize the neighborhood information of the nodes. KGAT [16] combines the user item interaction matrix with the knowledge graph in the Collaborative Knowledge Graph (CKG) embedding layer and obtains the graph item vector representation through embedding. Then, in the attention embedding propagation layer, the item representation is enhanced by passing the propagation vector back to neighboring multi hop nodes. By calculating the relationship weights based on the attention mechanism of the knowledge graph, the node vector representation is completed after aggregating information. Finally, in the prediction layer, the user click probability is calculated and normalized through vector calculation, and recommendation is achieved. KGIN [17] models each intention as a combination of relationships in the knowledge graph to achieve better modeling capability and interpretability. In addition, this method proposes a new information aggregation scheme that recursively integrates the relationship paths of remote connections. KGIN provides interpretability for predictions by identifying influential intentions and relationship paths.

B. *Contrastive Learning*

The contrastive learning approach [18,19] acquires node representations by differentiating between positive and negative examples. Initially, DGI [18] incorporated Infomax into graph representation learning, focusing on contrasting local with global node embeddings. Following this, GMI [20] proposed contrasting central nodes with their adjacent nodes, considering both node attributes and structural positions. In a similar vein, MVGRL [21] generates node and graph-level representations of neighborhoods and graph propagation from two distinct structural perspectives (including first-order graphs), and contrasts the encoded embeddings across these two views. More recently, HeCo [19] introduced learning node representations from both network pattern and meta path perspectives, conducting contrasting learning between them. KGCL [22] employs a KG augmentation scheme to mitigate noise in information aggregation. It also leverages additional supervisory signals from the KG enhancement process to guide cross-view contrastive learning, further suppressing noisy user-item interactions. However, KGCL only performs contrastive learning between the KG and user-item views, which not consider the complete semantic information in the CKE view.

III. Method

A. Preliminaries

In a recommendation scenario, we typically have historical user-item interactions such as purchases and clicks. Here, we represent this interaction data as a bipartite graph between users and items, defined as {(u, Y_{ui}, i) | u ∈ U, i ∈ I}, where U and I denote the sets of users and items respectively, and a connection Y_{ui} = 1 signifies an observed interaction between user u and item i; otherwise, Y_{ui} = 0.

In addition to user-item interactions, our model incorporates side information for items, comprising attributes and external knowledge that enriches item descriptions. This supplementary data involves real world entities interconnected through various relationships, effectively profiling each item. To bridge the gap between items in our primary dataset and entities within KG, we establish a mapping referred to as item-entity alignments, represented by the set A = {(i, r, e)|i ∈ I, e ∈ E, r∈ R}. Each pair (i, e) in A signifies that the item i corresponds directly to an entity e within KG, thereby integrating domain-specific knowledge into our recommendation framework.

The concept of a Comprehensive Knowledge Graphs (CKG) is introduced, merging user behaviors and item knowledge into a unified graph. Each user action is depicted as a triplet (u, Y_{ui}, i), signifying an 'Interact' relation between user u and item i when Y_{ui} = 1. Leveraging item-entity alignments, the user-item graph integrates smoothly with KG, forming a unified graph G = {(h, r, t)|h, t ∈ E', r ∈ R'}, where E' combines entities E from KG with users U (E' = E ∪ U), and R' expands relations R with Yui (R' = R ∪ {Y_{ui}}).

B. CKG Based Graph Attention Network

Firstly, the TransR [23] method is used to generate the embedding representations of CKE. Consider entity h, represented by N_h = {(h, r, t) | (h, r, t) ∈ G}, which denotes the set of triples with h as the head entity. To characterize the first-order connectivity structure of entity h, this model calculates a linear combination of h's neighborhood N_h:

$$E_{N_h}=\sum_{(h,r,t)\in N_h}\pi(h,r,t)e_t \tag{1}$$

$$\pi(h,\ r,\ t)=(W_re_t)^{\top}tanh((W_re_h+e_r)) \tag{2}$$

Where π(h,r,t) represents the weight parameters associated with the tail entity, and tanh is a non-linear activation function.

Following the aggregation of information for entity eh and its neighborhood combination representation e_{Nh}, we obtain $e_h^{(1)}$= f (e_h, e_{Nh}), where f serves as the aggregator. This paper further explores higher-order connection information by gathering signals propagated from higher-hop neighbors and concatenates multi-hop vectors to achieve a final global-level representation for users and items:

$$e_h^{(l)}=f\left(e_h^{(l-1)},e_{N_h}^{(l-1)}\right) \tag{3}$$

$$e_u^{glo}=e_u^{(0)}\,||\cdots||\;e_u^{(L)},\ e_i^{glo}=e_i^{(0)}\,||\cdots||\,e_i^{(L)} \tag{4}$$

C. Automatic Masking Mechanism

Firstly, calculate the different weights between project i and the entity e it is connected to in KG:

$$g\left(e,r_{e,i},i\right)=\frac{exp\left(LeakyReLU\left(r_{e,i}^{\top}W[x_e||x_i]\right)\right)}{\sum_{e\in N_i}exp\left(LeakyReLU\left(r_{e,i}^{\top}W[x_e||x_i]\right)\right)} \tag{5}$$

Then, noise is added to the attention weights g(e,$r_{e,i}$,i):

$$g'\left(e,r_{e,i},i\right)=g\left(e,r_{e,i},i\right)-log(-lo\,g(\epsilon));\epsilon\sim Uniform(0,1) \tag{6}$$

where ϵ is a random variable sampled from a uniform distribution. Treat the representation learned for item i from KG as one contrastive view, and consider the representation of item i' after adding random noise as another contrastive view.

$$x_i^{(l)}=e_i^{(l-1)}+\sum\nolimits_{(e,r,i)\in N_i}g(e,r_{e,i},i)x_e^{(l-1)} \tag{7}$$

$$x_{i'}^{(l)}=e_i^{(l-1)}+\sum\nolimits_{(e,r,i)\in N_i}g'(e,r_{e,i},i)x_e^{(l-1)} \tag{8}$$

The contrastive loss $\mathcal{L}_{noise}$ after adding random noise:

$$\mathcal{L}_{noise}=-log\frac{exp\left(s(x_i^{(l)},x_{i'}^{(l)})/\tau\right)}{\sum_{i=0}^{I}exp\left(s(x_i^{(l)},x_{i'}^{(l)})/\tau\right)} \tag{9}$$

Where s is the similarity function, and τ is the temperature parameter.

$$g\left(e,r_{e,i},i\right)=\begin{cases}g(e,r_{e,i},i)\in top\text{-}k\left(g\left(e,r_{e,i},i\right)\right)\\ 0, \qquad otherwise\end{cases} \tag{10}$$

Where LeakyReLU serves as the activation function, and W represents trainable parameters.

Secondly, unlike the random data augmentation scheme employed by the KGCL model, this model leverages the function g(e, $r_{e,i}$, i) to generate enhancement operators M_k^1 and M_k^2 for KG triples:

$$M_k^1=g\left(e,r_{e,i},i\right) \tag{11}$$

$$M_k^2=1-g\left(e,r_{e,i},i\right) \tag{12}$$

Where M_k^1 and M_k^2 ∈ {0,1}, and finally a specific selection is made for the neighborhood N_i of item i:

$$\eta_1(G_k)=\left((e,r,i)\odot M_k^1\right),\eta_2(G_k)=\left((e,r,i)\odot M_k^2\right) \tag{13}$$

Wherein, the masking vectors M_k^1 and M_k^2 indicate whether specific KG triples are selected during the sampling process.

D. Local Contrastive Learning

Firstly, data augmentation is performed on the user-item view by leveraging KG to enhance consistency among subgraphs:

$$c_i=s\left(f_k\left(x_i,\eta_1(G_k)\right),f_k\left(x_i,\eta_2(G_k)\right)\right) \tag{14}$$

Where f_k denotes the aggregator function, x_i represents the embedding of items in KG.

Following this, c_i is utilized to generate two masking vectors, M_u^1 and M_u^2, which are derived from a Bernoulli distribution [24], to perform data augmentation on the user-item interaction view:

$$\varphi(G_u)=\left(V,M_u^1\odot Y\right),\varphi(G_u)=\left(V,M_u^2\odot Y\right) \qquad (15)$$

Where V represents the set of nodes in the user-item interaction graph, and a random deletion is performed on the edge set Y within this interaction graph.

Following this, the nodes are encoded using the LightGCN [25] model:

$$e_u^{loc}=e_u^{(0)}+\cdots+e_u^{(L)},e_i^{loc}=e_i^{(0)}+\cdots+e_i^{(L)} \qquad (16)$$

$$\mathcal{L}_{loc}=\sum_{n\in V}-log\frac{exp(s(x_n^1,x_n^2)/\tau)}{\Sigma_{n'\in V,n'\neq n}exp(s(x_n^1,x_n^2)/\tau)} \qquad (17)$$

Where s is the similarity function, and τ is the temperature parameter, (x_n^1,x_n^2) are generated from the enhanced KG and the subgraph of user-item interactions mentioned above, $\mathcal{L}_{\mathrm{loc}}$ denotes the local contrastive loss function.

E. Global Contrastive Learning

Firstly, the node embeddings are fed into an MLP with one hidden layer:

$$z_c^{glo}=W^{(2)}\sigma\left(W^{(1)}e_c^{glo}+b^{(1)}\right)+b^{(2)} \qquad (18)$$

$$z_c^{loc}=W^{(2)}\sigma\left(W^{(1)}e_c^{loc}+b^{(1)}\right)+b^{(2)} \qquad (19)$$

Where W ∈ $R^{d\times d}$ and b ∈ $R^{d\times 1}$ are trainable parameters, and σ denotes the sigmoid function.

The sampling scheme for positive and negative examples is as follows: for any node in one view, the embedding of the corresponding same node learned in the other view serves as a positive example, while embeddings of all other distinct nodes are regarded as negative examples:

$$\mathcal{L}_{glo}=-log\frac{e^{s\left(z_c^{glo},z_c^{loc}\right)/\tau}}{\underbrace{e^{s\left(z_c^{glo},z_c^{loc}\right)/\tau}}_{\text{positive pairs}}+\underbrace{\sum_{k\neq c}e^{s\left(z_c^{glo},z_k^{glo}\right)/\tau}}_{\text{negative pairs within the view}}+\underbrace{\sum_{k\neq c}e^{s\left(z_c^{glo},z_k^{loc}\right)/\tau}}_{\text{negative pairs between the view}}} \qquad (20)$$

To combine the recommendation task with the self-supervised task, this paper adopts a multi-task training strategy to optimize the entire model. Firstly, the BPR [26] loss $\mathcal{L}_{\mathrm{BPR}}$ is constructed, therefor, the primary function after introducing the contrastive loss is as follows:

$$\mathcal{L}_{MVKGCL}=\mathcal{L}_{BPR}+\beta\left(a(\mathcal{L}_{\mathrm{loc}}+\mathcal{L}_{noise})+(1-a)\mathcal{L}_{glo}\right)+\lambda||\Theta||_2^2 \qquad (21)$$

Wherein, a and β are parameters respectively for regulating the weights of local-global contrastive loss and overall contrastive loss, while λ is the parameter that controls regularization. Details of the contrastive learning strategy are illustrated in Figure 2 below.

IV. Experiments

A. Dataset

The experiment selects two datasets, Amazon-Book (for product recommendation) and Yelp2018 (for business venue recommendation) [27], which exhibit different levels of interaction sparsity and KG characteristics. Detailed statistical information on the datasets is provided in Table I below.

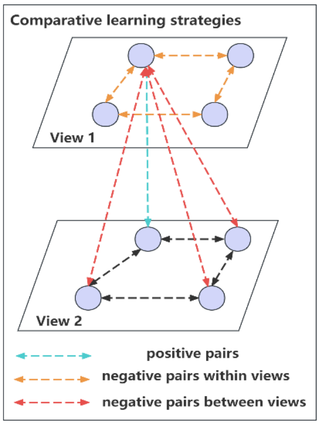

Comparative learning strategies
View 1
View 2
positive pairs
negative pairs within views
negative pairs between views

Fig. 2. Contrastive learning strategies

TABLE I. STATISTICS OF THE DATASETS USED IN EXPERIMENTS

		Amazon-book	**Yelp2018**
User-Item Interactions	***#Users***	70679	45919
	#Items	24915	45538
	#Interactions	846434	1183610
KG	***#Entities***	29714	47472
	#Relations	39	42
	#Triples	686516	869603

B. Evaluation Metrics

To evaluate the Top-N recommendation results of different models, we select two commonly used metrics for recommendation model, including Recall and Normalized Discounted Cumulative Gain (NDCG).

C. Evaluation Protocols

The recommendation model is implemented using the Pytorch deep learning framework, with the embedding vector dimension fixed at 64 for all methods. Model optimization is carried out with a learning rate of $1e^{-3}$ and a batch size of 2048. The initial Top-k is set to 20, with both Recall and NDCG metrics considered for model evaluation. This model also conducts grid search over relevant parameters: adjusting the Top-k value within {5, 10, 20, 50, 100}, setting local contrastive loss weights to {0, 0.2, 0.4, 0.6, 0.8, 1}, and contrasting learning weights to {1, 0.1, 0.01, 0.001} respectively.

D. Baselines for Comparison

In the experiments, we contrast the MVKGCL with several SOTA recommendation models, which can be divided into three categories：

- The first category comprises recommendation models based on traditional collaborative filtering methods: BPR [26]and GC-MC [28].
- The second category comprises recommendation models based on graph neural networks: LightGCN [25] and SGL[29].
- The third category comprises recommendation models based on knowledge graphs: CKE [8], RippleNet [11], KGCN[15], KGIN[17], CKAN[30], MVIN[31], and KGCL[22].

E. Performance Comparison with SOTA

Table II below presents the experimental results of all

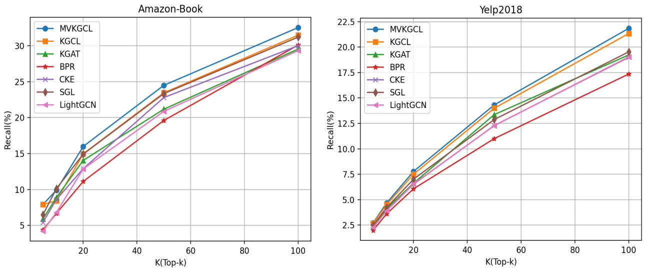


Fig. 3. The result of Recall@K in top-K recommendation

TABLE II. PERFORMANCE RESULTS OBTAINED

Model	Amazon-book		Yelp2018	
	Recall	***NDCG***	***Recall***	***NDCG***
BPR	12.44%	0.0658	5.55%	0.0375
GC-MC	10.33%	0.0532	5.35%	0.0346
LightGCN	13.98%	0.0736	6.82%	0.0443
SGL	14.45%	0.0766	7.19%	0.0475
CKE	13.75%	0.0685	6.86%	0.0431
RippleNet	10.58%	0.0549	4.22%	0.0251
KGCN	11.11%	0.0569	5.32%	0.0338
KGIN	14.36%	0.0748	7.12%	0.0462
CKAN	13.80%	0.0726	6.89%	0.0441
MVIN	13.98%	0.0742	6.91%	0.0441
KGCL	<u>14.96%</u>	<u>0.0793</u>	<u>7.56%</u>	<u>0.0493</u>
MVKGCL	**15.86%**	**0.0846**	**7.73%**	**0.0526**

models. Through observing the contrast experiments between model MVKGCL and the baseline models, the following observations are derived:

- Overall, in the experiments conducted on the Amazon-book and Yelp2018 datasets, MVKGCL demonstrates superior performance compared to other models. In terms of the Recall evaluation metric, MVKGCL achieves 15.86% and 7.73%, respectively, and for the NDCG evaluation metric, it reaches 0.0846 and 0.0526, respectively, surpassing the current SOTA model KGCL.

- In the Top-k recommendation task, our model MVKGCL surpasses the best baseline models in Recall@K metrics at multiple different K values. Among them, the joint recommendation methodologies (MVKGCL, MVIN, and KGCL) excel in recommendation effectiveness compared to embedding-based methods (CKE) and path-based methods (RippleNet). Specifically, on two datasets, MVKGCL enhances Recall by 3.5% to 4.4% and NDCG by 2.4% to 2.6% in contrast to CKE and RippleNet. The rationale behind this is that the model fully leverages the advantages of both embedding and path-based approaches, refining the depiction of entities and their relations through an iterative updating strategy, thereby compensating for the limitation of embedding methods in capturing higher-order semantic information. For more detailed outcomes, please refer to Figure 3 above.

- As a joint recommendation model, MVKGCL performs well, outperforming models such as KGIN, CKAN, and KGCL on both datasets. The main reasons for this include: firstly, the MVKGCL model, through global contrastive learning, comprehensively considers the complete structural information within the graph; furthermore, the automatic masking mechanism proposed in this paper adequately considers the varying attention weights between items and entities, selectively enhancing the data of KG.

F. Ablation Study of MVKGCL Framework

The experiment investigates the model's performance from the perspectives of KG and contrastive learning. Ablation experiments can verify the functions of different components of the model. MVKGCLw/o glo denotes the model variant without global contrastive learning, a component primarily utilizing an automatic masking mechanism to augment KG data. Meanwhile, MVKGCLw/o mask signifies the model version sans the automatic masking mechanism, which employs GAT for embedding learning on CKE graphs; subsequently, it leverages global contrastive learning to derive node embeddings enriched with semantic and structural information at the global level. Each component in the MVKGCL model contributes positively, with the complete MVKGCL model outperforming both MVKGCLw/o glo and MVKGCLw/o mask across evaluation metrics on two datasets. Furthermore, MVKGCLw/o glo consistently surpasses MVKGCLw/o mask in all metrics, highlighting the efficacy of employing attention weights to generate mask vectors for data augmentation of KG. The experimental results are shown in Table III below.

TABLE III. IMPACT STUDY OF MVKGCL MODEL VARIANTS

Model	Amazon-book		Yelp2018	
	Recall	***NDCG***	***Recall***	***NDCG***
MVKGCL	15.86%	0.0846	7.73%	0.0526
MVKGCLw/o glo	15.66%	0.0829	7.63%	0.0511
MVKGCLw/o mask	15.60%	0.0820	7.60%	0.0502

G. Impact of local-level contrastive loss weight a

Investigating the impact of local versus global contrast weights on model metrics. Specifically, to study the effect of the weight parameter b, the model varies a's value within the set {0, 0.2, 0.4, 0.6, 0.8, 1.0}, leading to the following observations: (1) For the Amazon-Book dataset, the model achieves its best performance when a=0.4; for the Yelp2018 dataset, the optimal performance is reached when a=0.2, indicating that at these points, a balance between local and global contrastive losses is achieved; (2) In the case of the Amazon-Book dataset, the worst performance typically occurs when a=0, highlighting the significance of the automatic masking mechanism. For the Yelp2018 dataset, poor performance is observed when the parameter a is either 0 or 1, suggesting that both levels of contrastive loss play a crucial role in the model's functioning. See Figure 4 below for further details.

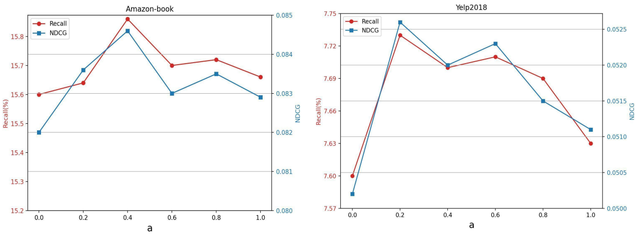


Fig. 4. Impact of local-level contrastive loss weight a

H. Impact of contrastive loss weight β

By adjusting the weights of contrastive learning, we explore the role contrastive learning plays in the model to uncover the significance of contrastive loss during multi-task training. Specifically, we vary the parameter β within the set {1, 0.1, 0.01, 0.001}, observing performance metrics across different datasets. The experimental results indicate that the model performs best when the parameter β is set to 0.1. The primary reason for this improvement is the adjustment of the contrastive loss to a level comparable with the recommendation task loss, thereby enhancing the model's performance. Details are provided in Table IV below.

TABLE IV. IMPACT OF CONTRASTIVE LOSS

	Amazon-book		**Yelp2018**	
	Recall	***NDCG***	***Recall***	***NDCG***
β=1	15.63%	0.0823	7.70%	0.0513
β=0.1	15.86%	0.0846	7.73%	0.0526
β=0.01	15.58%	0.0818	7.69%	0.0511
β=0.001	15.40%	0.0806	7.67%	0.0510

I. Vector Embedding Representation

To evaluate whether the contrast mechanism affects the performance of representation learning, this paper employs SVD decomposition to embed items into a two-dimensional space. As shown in Figure 5, this work contrasts the visualization results of MVKGCL, MVKGCLw/o glo, and MVKGCLw/o mask on the Amazon-book dataset. The following observations can be drawn from the figure below:

- The item node embeddings generated by MVKGCLw/o mask are mixed to some extent, while those produced by MVKGCLw/o glo fall into a narrow cone shape. In contrast, the node embeddings generated by the MVKGCL model exhibit a more diverse distribution: specifically, they are distributed more evenly and sparsely, thereby capable of representing different node feature information. This indicates that the MVKGCL model has superior capabilities in representation learning and mitigating representation degradation.

- By contrasting MCCLK and its variants, it is observed that removing the auto-mask or the global contrastive learning component makes the embedding representations less distinguishable. This evidence supports that the MVKGCL model enhances the effectiveness and robustness of representation learning.

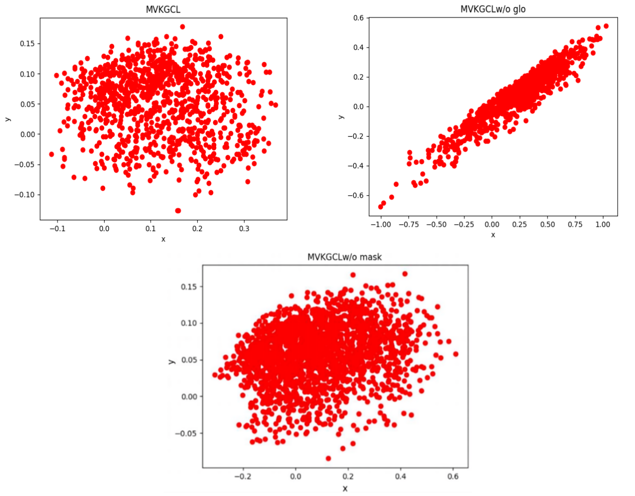


Fig. 5. Project Embedding Representation in Amazon-book

V. CONCLUSION AND FUTURE WORK

The work proposes a recommendation model MVKGCL based on KG and contrastive learning: firstly, an automatic masking mechanism is introduced to augment the data in KG; secondly, by employing graph attention neural networks, the complete structural information within CKG is mined. The node embeddings learned from this process are then globally contrasted with node embeddings obtained through local contrastive learning, fully exploiting the structural and semantic information within KG. Experiments constructed demonstrate that MVKGCL outperforms other existing models in terms of performance.

In future work, to address the issue of inadequate exploitation of structural views by the model, a new paradigm of graph attention neural networks will be considered for feature optimization of structural views. This enhancement aims to further elevate the model's performance.

ACKNOWLEDGMENT

The authors are immensely grateful to the anonymous reviewers and editors for their diligent efforts. Their valuable comments and constructive suggestions have significantly enhanced the quality of this work. We also extend our gratitude to the creators of the contrastive algorithms for generously sharing their source codes. This research was made possible through the financial support of the National Natural Science Foundation of China (62341607) and the Key Projects of the National Language Commission (ZD145-58).

REFERENCES

[1] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In WWW. 689–698.

[2] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In Recsys. 240–248.

[3] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In KDD. 950–958.

[4] Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu, and Huajun Chen. 2018.Label-free distant supervision for relation extraction via knowledge graph embedding. In EMNLP. 2246–2255.

[5] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019.Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In WWW. 151–161.

[6] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In AAAI.

[7] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In AAAI.

[8] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016.Collaborative knowledge base embedding for recommender systems. In SIGKDD.353-362.

[9] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019.Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In WWW. 151-161.

[10] Wang Z, Zhang JW, Feng JL, et al. Knowledge graph embedding by translating on hyperplanes. Proceedings of the 28th AAAI Conference on Artificial Intelligence. Quebec City: AAAI, 2014. 1112-1119.

[11] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie,and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In CIKM. 417-426.

[12] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Ok sana Yakhnenko. 2013.Translating embeddings for modeling multi-relational data. In Neural Information Processing Systems (NIPS). 1‑9.

[13] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018.Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In SIGKDD. 1531‑1540.

[14] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018.Heterogeneous information network embedding for recommendation. IEEE Transactions on Knowledge and Data Engineering (2018), 357‑370.

[15] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019.Knowledge graph convolutional networks for recommender systems. In WWW. 3307‑3313.

[16] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019.Kgat:Knowledge graph attention network for recommendation. In SIGKDD. 950‑958.

[17] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning Intents behind Interactions with Knowledge Graph for Recommendation. In WWW. 878‑887.

[18] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. ICLR (Poster) (2019), 4.

[19] Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2021. Self-supervised Hetero geneous Graph Neural Network with Co-contrastive Learning. arXiv preprint arXiv:2105.09111 (2021).

[20] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph representation learning via graphical mutual information maximization. In WWW. 259‑270.

[21] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In ICML. PMLR, 4116‑4126.

[22] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge Graph Contrastive Learning for Recommendation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. Association for Computing Machinery, New York, NY, USA, 1434‑1443.

[23] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In AAAI.

[24] Albert W Marshall and Ingram Olkin. 1985. A family of bivariate distributions generated by the bivariate Bernoulli distribution. J. Amer. Statist. Assoc. 80, 390(1985), 332‑338.

[25] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In SIGIR. 639‑648.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. arXiv (2012).

[27] Ding Zou, Sen Zhao, Wei Wei, Xian-ling Mao, Ruixuan Li, Dangyang Chen, Rui Fang, and Yuanyuan Fu. 2023. Towards Hierarchical Intent Disentanglement for Bundle Recommendation. IEEE Transactions on Knowledge and Data Engineering(2023).

[28] Rianne van den Berg, Thomas N Kipf, and Max Welling.2017. Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263.

[29] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In SIGIR.726‑735.

[30] Ze Wang, Guangyan Lin, Huobin Tan, Qinghong Chen, and Xiyang Liu. 2020. CKAN: Collaborative Knowledge-aware Attentive Network for Recommender Systems. In SIGIR. 219‑228.

[31] YChang-You Tai, Meng-Ru Wu, Yun-Wei Chu, Shao-Yu Chu, and Lun-Wei Ku. 2020. MVIN: Learning Multiview Items for Recommendation. In SIGIR. 99‑108.

Learning based Approach for Optimizing Microservices Configuration Parameters

Sulochan Naik , Meenakshi D'Souza
International Institute of Information Technology Bangalore
Bangalore, India
sulochan.naik@iiitb.ac.in,meenakshi@iiitb.ac.in

Abstract

Adoption of microservices has increased exponentially worldwide. Despite higher adoption, there is a lack of work on possible auto tuning of configurable parameters for various optimisation in terms of cost, resources and other aspects. We propose a novel way to find the best possible configuration for a service based on certain parameters. Each service is modeled mathematically by considering the various configurable parameters that a service can have. We then use learning methodologies to optimise on the cost and find the best parameters for optimised on the cost without affecting the service, thereby cutting down on over provisioning of the configuration. Our in-house implementation and experimentation on Samsung owned microservices achieved an optimization in terms of cost by $1593 per month. When compared with the benchmark commercial solution called Kubecost, our solution outperformed by 33% on an average for CPU and 27% for memory across 12 microservices over a month of data.

Index terms— Microservices, Configurable Parameters, Bayesian Optimization, Objective Function, Surrogate Function

1 Introduction

Adoption of microservices has become a new industry standard owing to several of its inherent properties and the resulting advantages. According to the report [11], microservices had an adoption rate of 37% worldwide and is increasing exponentially. The need for better maintenance and optimization would be the key with increased adoption. Typically parameters of a service are configured blindly without taking the nature of a service and its behaviour into consideration. A service might just be a proxy where no computation heavy processes exists or there could be a computationally intensive service. Often configurable parameters are over provisioned by blindly setting a value inherited from another service.

In this paper, we propose a novel way to provide optimal configurable parameters thereby reducing cost of overall service without affecting the functionality. An important thing to consider is that there is no liberty in live production systems to change configuration, deploy, observe and penalise or award the changed configuration. If a configuration doesn't work, we need to change and redeploy. Since we don't have the option to re-configure on live production systems and the available data set to handle this problem is limited, we propose a Bayesian Optimization algorithm [13] which suits such kind of live production systems.

Our methodology was implemented and tried on various in-house Samsung owned microservices meant for achieving inter-connectivity and interoperability among the smart devices and appliances for memory configuration. Our methodology achieved optimization in terms of cost by 1593 USD per month, by decreasing the CPU and memory configuration for an optimized number of running instances. We also illustrate how our approach saves important configurable parameters when compared with a commercial benchmark, Kubecost [7].

Rest of the paper is organized as follows. Section 2 deals with the background on microservices and Bayesian Optimization. Section 3 contains related work in in the area of optimization of the various parameters of microservices. Section 4 explains our methodology which is followed by results and analysis in Section 5. We conclude in Section 6.

2 Background

2.1 Microservices

The microservices architectural style is an approach to develop a single application as a suite of small services, each running in its own process and communicating us-

DOI reference number: 10.18293/DMSVIVA2024-133

ing lightweight mechanisms, often an HTTP resource API [4, 10]. These services are built around business capabilities and are independently deployable by a fully automated deployment machinery [4, 10]. There is a bare minimum of centralised management of these services, which may be written in different programming languages and use different data storage technologies [4, 10]. Decentralisation and Componentization are major characteristics of microservices.

In a microservices architecture, each service, apart from interacting with other services, also interacts with various resources like databases, queues or message passing services and other consuming application. Configuration management in microservices involves defining a consistent way to configure the connections and other operating parameters of each service. In an ideal state, the configurations should be adapted without the applications needing a re-start. This is the main problem considered in this paper.

2.2 Bayesian Optimization

Bayesian optimization is a methodology to find optimal values for expensive objective functions and it has proven success in various fields including Sciences, Engineering and beyond [9, 5, 6]. It is a sequential design strategy for global optimization where functions are not known or are too expensive to formulate. Bayesian Optimization builds a probability model of the objective function and uses it to select hyper-parameter to evaluate in the true objective function. [9, 5, 6]. There are various terminologies associated with this optimisation technique. A *true objective function* is the actual function which depicts your service. *Surrogate functions* approximate true objective functions and are used initially to start with. An *acquisition function* is used further and the next hyper-parameter of choice is where the acquisition function is maximized. After using an acquisition function to determine the next hyper-parameter, the true objective function score of this new hyper-parameter is obtained. Since the surrogate model has trained on the (hyper-parameter, true objective function score) pairs, adding a new data point updates the surrogate model. The above steps are repeated for several iterations (until maximum iterations are reached or memory is exhausted) to be as close to true objective function as possible.

3 Related Work

So far, the research focus has been on monitoring various aspects of microservices and suggesting certain cost optimizations based on simple techniques like usage, average, minimum, maximum etc. Kubecost is able to associate a cost in a given time window to each of the main Kubernetes abstractions: from the Pod to the Service, from the Deployment to the Namespace [7, 2, 3]. It suggests an optimization based on the average usage with utilization capped to 65% of usage. Cost Optimization Strategies for AWS Infrastructure [12] rely on a heuristics approach based on monitoring of various cost explorers rather than a data-driven approach. Optimization of resource provisioning costs in cloud computing [2] suggests various optimization techniques by exploring various provisioning of services like reserved, spot etc which are again based on the acumen gained but not the data driven approach. There is no focus on auto tuning of configurable parameters using data-driven approaches and thereby optimizing services based on their nature. Our solution focuses on achieving this by replacing the heuristics based approach or simple techniques used so far with a technique based on Bayesian optimization. The above-mentioned techniques will not work effectively for auto tuning as they work without considering the nature of a service and properties like whether service requires high computation, resources etc.

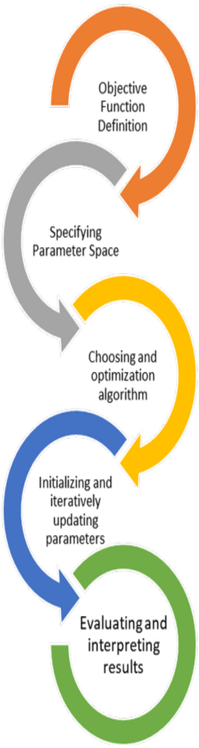

Objective Function Definition
Specifying Parameter Space
Choosing and optimization algorithm
Initializing and iteratively updating parameters
Evaluating and interpreting results

Figure 1: Various steps for our model

4 Methodology for Optimizing the Configuration Parameters

Figure 1 gives a description of our proposed methodology. Initially, we consider as input, any micro service with all the configurable parameters where optimization is required and perform various steps required for optimization which includes set of parameters under considerations and ground truth of sample distribution. A loss function is calculated with training data set and an appropriate algorithm . Starting with the definition of the objective function, the various parameters are specified in the second step and the Bayesian optimization algorithm is applied. The iterations of the algorithm through the various steps as highlighted in Section 2.2 are completed to obtain the desired results containing the configuration parameters. Our experiments indicate that the parameters are indeed optimized when compared to other benchmarks.

We now present the details of the algorithm. Mathematically, we would like to compute the following function

$$\lambda^* = argmin_{\lambda \in \Lambda} E_{x \in \mathcal{G}_x} [\mathcal{L}(x, \mathcal{A}(D_{training-data}))] \quad (1)$$

where Λ is the set of configuration parameters that we would like to optimize (e.g., CPU, memory etc.), $\mathcal{G}$ is the ground truth of the sample distribution, and x is the underlying machine learning algorithm that operates over the training data set $\mathcal{A}_\lambda$. The algorithm that will compute this equation will minimize the loss function $\mathcal{L}$.

Table 1 provides the list of various configurable and derived parameters that are considered as input. There are various categories under which we can have various configurable parameters for a service. CPU and memory are basic configuration that each service would have. CPU and memory desired capacity are configured while we can optimise based on derived parameter that is utilisation percentage found through various monitoring tools. Similarly scaling policy, storage, cache etc are other categories which are configured and we found out whether they are over provisioned or under provisioned based on derived/measurable attributes

An objective function is nothing but a function which determines our service. Defining true objective function is difficult for our scenario. Reason being there are lot of parameters to consider and its complex to arrive at true objective function with limited data that we have. Hence, we start with a surrogate objective function based on data we have. We use following steps for our model to function:

- We initiate a surrogate model and an acquisition function. Surrogate model is kind of false objective function like Gaussian which is improved to make it close to true objective function by using acquisition functions which picks next best candidate based on certain rules [1]

- Then for each iteration, we find the parameter x^* where the acquisition function is maximized. The acquisition function is a function of the surrogate model, which means it is built using the surrogate model instead of the true objective function.

- Obtain the objective function score of $x^*$to check the performance of this point

- Next, we have to include the (parameter x^*, true objective function score)in the history of other samples

- Next, we have to train the surrogate model using the latest history of samples and these steps to be repeated until the max number of iterations is reached

5 Results and Analysis

Model explained earlier was applied on various services owned by Samsung for it's inter device (Internet of Things) operations. Data required to feed the model were taken from couple of places. Configurable parameters' values were taken from code repository and derived parameters were captured from Datadog [8]. Datadog is a tool which is used to monitor various parameters that are fed from services. All the running instances feed data to this tool which can be sought whenever required. Experiments were performed independently on various parameters. CPU utilisation was captured with number of instances and this particular scenario was optimised and ideal value of CPU for minimal number of instances was calculated. Model was then built for higher dimension to consume cpu and memory together and suggest optimal values for the chosen parameters.

Table 2 provides the results obtained for CPU from 12 different microservices in comparison with benchmark. It can be clearly seen from the Figure 2 as well that our solution outperforms benchmark solution for every services. Overall there is an average improvement of 33.32% per service as compared to benchmark solution. There is also a cost savings of roughly 1593 USD per month accross 12 services calculated based on cpu savings against actual cost of these services.

Table 3 provides the results obtained for memory from 12 different microservices in comparison with benchmark. It can be clearly seen from the Figure 3 as well that our solution outperforms benchmark solution for every services. Overall there is an average improvement of 27.83% per service as compared to benchmark solution.

Figure 4 shows one of the sample result with various iterations (20 iterations were chosen based on standard) through which objective function was improved and model suggested optimised default value for CPU.

6 Conclusion

In this work, we proposed a model which will provide optimal values for various configurable parameters based on

Table 1: Common Parameters

S.No	Configurable Parameters	Derived Parameters
1	CPU desired capacity	CPU Utilization
2	Memory desired capacity	Memory Utilization
3	Language specific RAM (JVM)	Heap/Threads
4	Scaling Policy (threshold)	Number of Pods
5	Queue	Queue size per event
6	Cache	Cache Hit/Miss
7	Storage Parameters	Record size

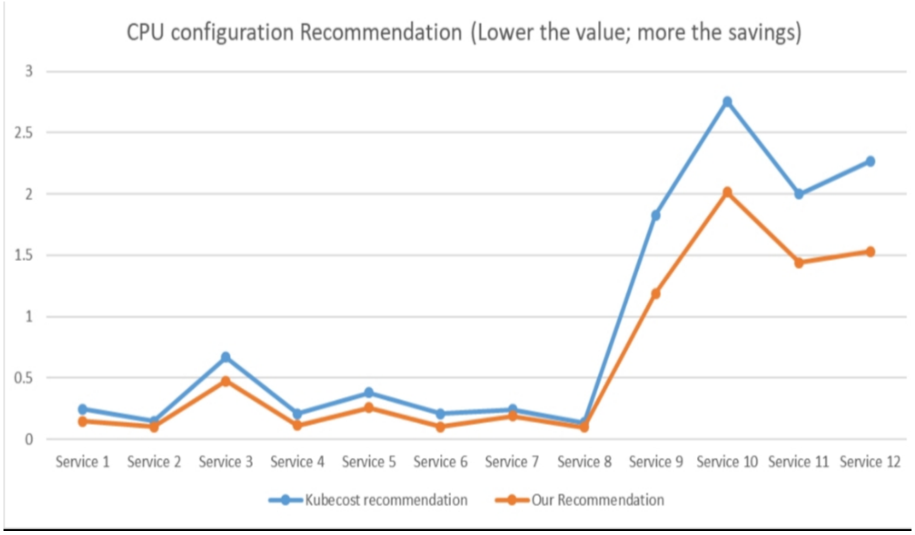


Figure 2: CPU configuration recommendation

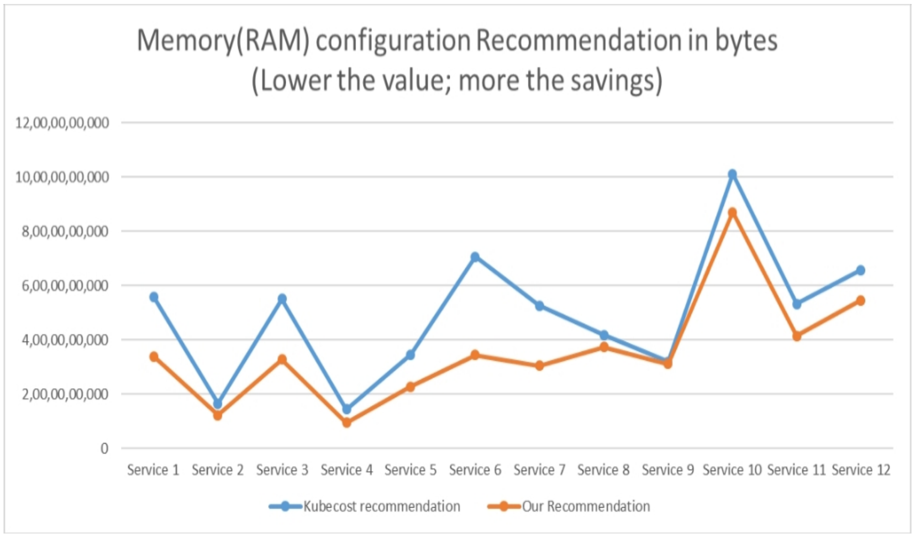


Figure 3: Memory configuration recommendation

Table 2: CPU Result comparison with Kubecost benchmark soluton

Service	CPU cores requested	Kubecost recommended CPU cores	Recommendation based on our model	Percentage improvement	Savings (Dollar per month)
Service 1	1	0.244	0.146	40.2%	74.02
Service 2	1	0.147	0.101	31.3%	41.76
Service 3	2	0.669	0.472	29.3%	97.74
Service 4	0.7	0.207	0.114	44.9%	36.2
Service 5	1	0.379	0.257	32.2%	30.4
Service 6	3	0.206	0.1	51.5%	305.4
Service 7	1	0.24	0.187	22.1%	28.06
Service 8	2	0.134	0.099	26.2%	92.06
Service 9	2	1.829	1.188	35.04%	8.64
Service 10	3	2.757	2.016	26.88%	40.17
Service 11	3.1	1.998	1.44	27.92%	798.96
Service 12	2	2.27	1.53	32.4%	40.07

Table 3: Memory(RAM) Result comparison with Kubecost benchmark solution

Service	Kubecost recommended RAM in bytes	Recommendation based on our model	Percentage improvement
Service 1	5,58,66,98,240	3,38,78,90,236.32	39.36%
Service 2	1,65,23,57,908	1,22,57,10,064.8	25.82%
Service 3	5,49,96,04,677	3,27,06,54,417.168	40.53%
Service 4	1,44,29,45,182	94,86,84,140.424	34.2%
Service 5	3,45,16,36,185	2,27,72,61,561.35	34.02%
Service 6	7,06,02,62,597	3,44,39,96,994.51	51.2%
Service 7	5,26,51,43,336	3,04,52,70,026.47	42.2%
Service 8	4,17,90,92,087	3,72,72,36,137.75	10.8%
Service 9	3,18,90,32,173	3,10,15,24,558.77	2.74%
Service 10	10,11,09,00,382	8,69,53,74,328.52	14.02%
Service 11	5,31,58,32,911	4,14,63,49,670.58	22.04%
Service 12	6,56,24,15,853	5,44,68,05,157.99	17.03%

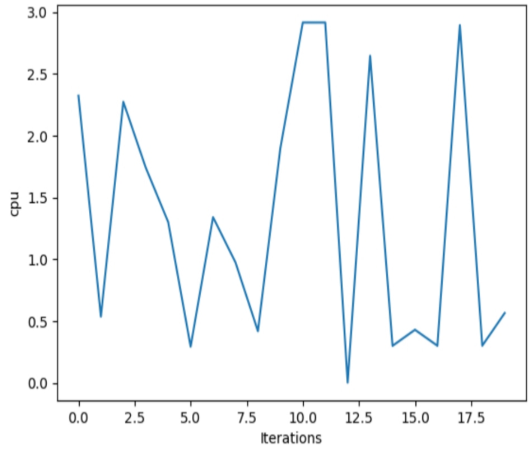


Figure 4: Results for various iterations and improvement

nature of service and thereby save cost as high as 1593 USD per month. In addition, this model will also help to auto tune parameters during sudden increase/decrease in load, predict future values based on history and finding maximum and minimum value that a particular metrics can have for a standardised configuration set. Our model is compared with one of the benchmark solutions where our model outperforms the other one. This model will save lot of manual effort and cost for all the cloud services that exist in this era.

References

[1] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[2] S. Chaisiri, B.-S. Lee, and D. Niyato. Optimization of resource provisioning cost in cloud computing. *IEEE transactions on services Computing*, 5(2):164–177, 2011.

[3] F. Cicchiello. *Analysis, modeling and implementation of cost models for a multi-cloud Kubernetes context*. PhD thesis, Politecnico di Torino, 2021.

[4] M. Fowler. Microservices, 2014.

[5] R. Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

[6] P. Hennig, M. A. Osborne, and H. P. Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022.

[7] kubecost. kubecost, 2019.

[8] A. Mckaig and T. Khan. How the metrics backend works at datadog. 2022.

[9] J. Mockus and J. Mockus. *The Bayesian approach to local optimization*. Springer, 1989.

[10] S. Naik and M. D'Souza. Detection of faults in microservices using petri nets. In *Proceedings of the 16th Innovations in Software Engineering Conference*, pages 1–5, 2023.

[11] Organizations' adoption level of microservices worldwide in 2021, 2021.

[12] L. Tammik. Cost optimization strategies for aws infrastructure. *Integrated Journal of Science and Technology*, 1(2), 2024.

[13] W. Wang. Bayesian optimization concept explained in layman terms. *Towards Data Science*, 2020.

Uncovering the Effects of Quantum Computing on Software Engineering: A Systematic Mapping

Valter Vieira de Camargo[*], Renato Bueno[*‡], Caio O. W. G. Cadini [*‡], Guilherme d. S. Wisniewski[*‡]
Augusto d. S. G. Vaz[*‡], Caio U. Sampaio[*‡], Vanderlei de Brito Junior[*‡] and Daniel San Martín[†]
[‡]Tutorial Educational Program (PET-BCC/UFSCar), [*]Federal University of São Carlos (UFSCar), SP, Brazil,
[†]EIC-Universidad Catolica del Norte, Coquimbo, Chile

***Abstract*—Quantum computing has attracted the attention of researchers and practitioners in recent years. It presents a completely new way of thinking about computation, offering much faster problem-solving performance compared to classical computing. As with any new paradigm, the emergence of quantum computing requires us to revisit many established concepts and methodologies—this is particularly true for software engineering. Established methodologies, quality attributes, and processes provided by software engineering need to be reviewed in light of quantum computing. Therefore, in this paper, we present a systematic mapping (SM) of how quantum computing has affected software engineering. The goal is to understand what researchers have accomplished regarding four software engineering topics: testing and quality, reengineering/modernization, modeling and processes, and development platforms. We considered the most important digital libraries up to June 2023.**

***Index Terms*—mapping review, quantum computing, software engineering**

I. Introduction

Quantum computing has emerged as one of the most promising technologies of the 21st century. It relies on the power of qubits, the fundamental units of quantum information. Unlike classical bits that can represent either 0 or 1, qubits can exist in a superposition of states, enabling quantum computers to process vast amounts of information simultaneously. This capability allows efficiently solve previously intractable problems [1] [2].

Software engineering concerns the development of techniques, methodologies, and tools aimed at assisting the development of high-quality software [3]. Whenever a new development paradigm emerges, software engineering needs to be reviewed to accommodate new features and provide techniques and methods more suitable for the new context.

Similar to what happened with the advent of object-oriented programming, aspect-oriented programming, and other paradigms, we are currently facing an even more drastic change as a new form of computation presents itself. In this regard, various aspects of software engineering need to be revisited in the light of quantum computing, including development processes, code quality metrics, modeling techniques and languages, and more.

While software engineering plays a pivotal role in the creation and maintenance of complex software systems, quantum computing presents a revolutionary potential for efficiently solving previously insurmountable problems. As these two fields progress, the pursuit of synergy between them becomes increasingly pertinent.

DOI reference number: 10.18293/DMSVIVA2024-090

Nevertheless, despite the presence of numerous research groups worldwide dedicated to various aspects of applying software engineering to quantum computing, a comprehensive systematic mapping (SM) that synthesizes these efforts remains elusive. The advent of quantum computing has introduced novel challenges, necessitating a reevaluation of established software engineering principles. Consequently, researchers must reassess fundamental concepts, design methodologies, and software development practices.

This paper presents a SM providing an overview of four critical aspects of software engineering: i) testing and quality; ii) reengineering/modernization; iii) modeling and iv) processes and development platforms/IDEs. This is a systematic mapping that aims to complement a quasi-systematic-mapping we have previously developed [4]. Here, we delineate the methodology employed to conduct this SM and offer an analysis of the principal findings and identified trends.

The paper is structured as follows. Section II provides the background information, followed by Section III detailing the SLR methodology. Section IV presents the results, and Section V concludes the findings.

II. Quantum Computing and Hybrid Systems

In classical computing, the most basic unit of information is the so-called *bit*, represented by two possible values, 0 and 1. In this scenario, data is processed and manipulated using a binary approach, working with one value at a time. However, with the emergence of quantum computing, information is currently portrayed through a fundamental unit known as the *qubit*, used to depict data and perform operations with increased speed and efficiency, leveraging its intrinsic characteristic of superposition [5].

A *qubit* can be conceptualized as a mathematical abstraction capable of existing in two distinct states, denoted by $|0\rangle$ and $|1\rangle$. In contrast to classical computing, where a bit can only hold either the value 0 or 1 at a given time, a *qubit* can exist in what is known as a superposition of the states $|0\rangle$ and $|1\rangle$, which in its most general form is formalized as Figure 1, left formula. Note that α and β are complex numbers satisfying $|\alpha|^2 + |\beta|^2 = 1$. Consequently, we can represent the state of a *qubit* as a normalized vector in a complex vector space of dimension 2, as illustrated in Figure 1, right formula [5].

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad |0\rangle = \begin{pmatrix}1\\0\end{pmatrix},\ |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

Fig. 1: A qubit and Vectorial representation of states

The superposition allows the performance of calculations and operations with the qubit in both states simultaneously, enabling a significant leap compared to the processing time of classical computers. However, following the superpositional phenomenon, the qubit collapses after measurement, adhering to only one of the possible information states. In this regard, it becomes essential to manipulate the qubit in a way to avoid collapse, in this case, through the use of quantum gates.

Among the quantum gates, special emphasis is given to the Hadamard and CNOT gates. The first one is responsible for putting the input qubit in a balanced superposition state as output, where there are equal chances of the qubit collapsing to $|0\rangle$ or $|1\rangle$. The second one takes as input a control qubit and another qubit; if the control qubit is in superposition (equal chances of collapse to both quantum states), it is said that the two input elements are now entangled.

Currently, there has been a growing number of different hardware implementations in the field of quantum computing [6]. The differences brought by quantum computing are quite fundamental for the current understanding of computing, requiring that various aspects that were once established be revisited in the light of this new form of computation. The emergence of quantum programming languages, for example, demands a reconsideration of various aspects of software engineering, such as software testing, development processes, source code metrics, modeling techniques, and more.

Hybrid quantum applications, typically combine both quantum algorithm implementations (QAIs) and classical programs [7]. These hybrid applications often involve multiple QAIs, such as clustering followed by classifier training, alongside classical programs responsible for tasks like data loading and visualization [8]. Even within a single QAI, hybridity persists, as quantum programs (QP) are integrated with classical ones. The structure of a gate-based QAI typically involves classical programs handling preprocessing tasks, like state preparation circuit generation, executed on classical computers [9]. Quantum programs, represented as quantum circuits, are then executed on a quantum computer, with pre- and post-processing tasks typically managed by classical programs [10].

III. Methodology of the Systematic Mapping

SM involve the selection, analysis, and categorization of scientific works to provide a broader overview of a particular area [11]. The specific goal of this SM is providing a detailed analysis on how quantum computing has affected four software engineering topics: Software Testing and Quality (TQ); Software Reengineering/Modernization (RM); Software Modeling (M) and Software Processes and Development Platforms (PP). This section is divided into 3 subsections: Protocol, Conduction and Data Extraction. Protocol is the phase responsible for detailing the initial preparation of the SM. In the second subsection, called Conduction, the focus is on how the protocol was put in practice. In the Data Extraction section we extract some important data from the final set of papers.

*("Software Model" **OR** "Software Engineering" **OR** "Software Development" **OR** "Software Lifecycle" **OR** "Software Development Methodologies" **OR** "Software Project Management" **OR** "Testing" OR "Design Pattern" **OR** "Reengineering" **OR** "Reverse Engineering" OR "Metrics" **OR** "Service-Oriented") **AND** ("Quantum Software" **OR** "Quantum Programming" **OR** "Quantum Computing" **OR** "Quantum Software Development" **OR** "Quantum Subroutine" **OR** "Quantum Program")*

Fig. 2: Base Search String

A. Protocol

We have defined four research questions (RQs) covering four software engineering topics:

- **RQ1**: How has QC impacted software testing and quality? This RQ aims to identify proposed methods for testing quantum software and topics related to quantum metrics and bugs.
- **RQ2**: How has QC influenced software reengineering and modernization practices? This RQ seeks to analyze the treatment of reverse engineering and refactorings within this context.
- **RQ3**: How to model quantum software? This RQ aims to investigate the impact of QC on modeling languages and the granularity level used.
- **RQ4**: How have development processes and platforms been shaped by QC? The goal is to assess the current maturity of processes and platforms for QC.

Based on the RQs, we created the Base Search String shown in Figure 2, that was adapted to the syntax of each digital library. The upper part includes terms related to software engineering and the lower part covers the quantum terms.

The digital libraries we have used in the searches were the following: ACM, Science Direct, IEEE Xplore and Scopus.

Another important task of the protocol is the definition of inclusion and exclusion criteria for filtering the studies. This is necessary to refine the initial set of primary studies obtained. These criteria were applied throughout the filtering stages to converge towards a final set of primary studies that truly met our goals. The inclusion criteria (IC-#) and the exclusion ones (EC-#) were the following: **[IC-1]** The study is published in English; **[IC-2]** The content of the study is related to the topic being analyzed; **[EC-1]** The study is not available; **[EC-2]** The study is duplicated; **[EC-3]** The study is not primary study surveys, systematic reviews/mappings, talks, proceedings, etc); **[EC-4]** The content of the study is not related to the theme or it is too superficial; **[EC-5]** The content of the study was updated in a next more-complete version.

B. Conduction

Figure 3 presents all the steps we have followed for filtering the initial set of 1299 recovered studies. This figure details the stages of the filtering process and the number of papers being removed as the exclusion criteria are applied. In the first step, called duplicate removal, 265 papers were removed, resulting in a total of 1034. In the second step, papers that aren't primary studies were removed resulting in a total of 936 papers. In the third step, a reading process of titles and abstracts was carried out to identify papers that did not meet the desired criteria. In this step, 625 primary studies were then removed. In the last stage of the process, the papers were fully read, in this stage, 257 papers were removed, resulting in a final set of 53 papers.

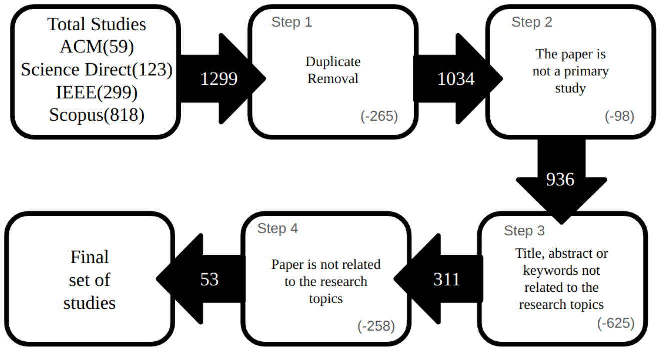


Fig. 3: Steps of the Filtering Process

The papers of the final set can be seen in Tables I and II. The first table shows journal papers and the second one conference papers. In both tables, there is an identification we have created to facilitate the reference to the paper. Papers identified with P# were published in journals and with C# in conferences. Regarding the conferences, we have used just the acronym because of space limitations. In the second column is the title, in the third the authors, in the fourth the publication venue, and in the fifth the publication year. In both tables, the papers are classified chronologically, from the oldest to the newest.

The final column in both tables categorizes the papers based on our objectives to address the research questions. The acronyms represent the following: PP = Processes and Platforms; TQ = Testing and Quality; RM= Reengineering/Modernization, and M = Modeling. In cases where the paper's content is related to more than one topic, we categorize it based on the area where it provides the most significant contribution.

C. Data Extraction

Figure 4 shows the classification of final set according to the classification proposed by Wieringa *et al.* [12]. 50.9% of the papers were classified as Proposal of Solution, which are proposals of approaches or techniques but without a complete validation. 22.6% were classified as Validation Research, which concentrates on the properties of a proposal that has not been implemented. 15.1% were classified as Evaluation, which are papers that present assessments or technique implementation in practice. 9.4% were classified as Opinion Papers, which express an opinion. The category with the least number of papers was Personal Experience Papers, with only one representative (1.9%), which emphasized what and not why in the authors' experience. We believe Proposal Of Solution represent the highest value in consequence of the novelty of the QC theme. Figure 5 depicts the evolution of paper publications by year and topic.

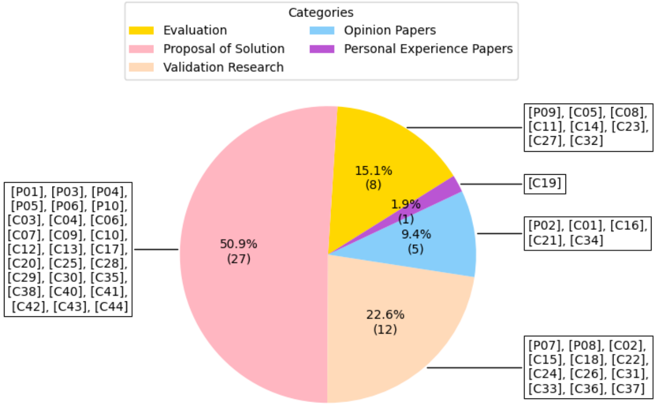


Fig. 4: Classification According to Wieringa *et al.* [12]

A clear trend emerges: there has been a steady increase in publications over the past four years (2018 – 2022). This highlights the growing interest in software engineering for quantum computing, indicating its status as a prominent and dynamic field captivating the software engineering community. Analysis by research topic shows a notable focus on Testing and Quality (TQ), while Modeling (M) and Reengineering/Modernization (RM) have received less attention. This trend could be explained due to the inherent complexities involved in testing quantum software.

For example, the superposition property allows qubits to exist in multiple states simultaneously, complicating testing tasks due to the unpredictability of qubit behavior until measured. Therefore, it is plausible that significant effort is being directed towards developing approaches for testing quantum software to ensure the reliability of quantum systems.

On the other hand, as the technology has not yet reached a level of maturity where it can reliably outperform classical computing for a wide range of practical tasks, many practitioners are hesitant to invest significant resources into a technology that is still evolving. As a consequence, reengineering and modernization tasks still are not strongly required. However, with growing acceptance anticipated in the near future, we foresee a surge in interest in this research domain.

IV. Overview of Papers and Answers for RQs

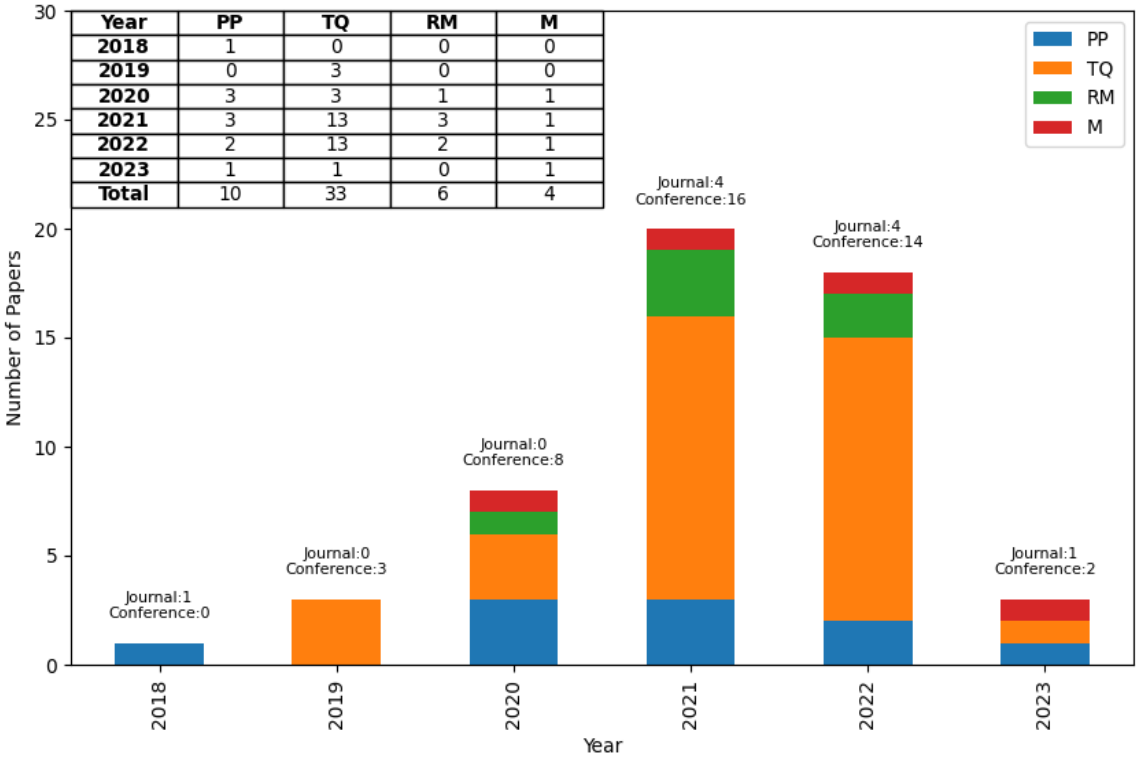
This section has four subsections; each one dedicated to address one RQ. At the end of each subsection one can find the answer to the RQ.

RQ1 - How has QC impacted software testing and quality?

As this RQ involves two related topics (testing and quality), we decided to break it down in two sub-RQs. One focusing no testing and another addressing quality.

RQ1.1 - How has QC impacted software testing?
To answer this RQ we considered twenty papers divided into the following sub-categories: 1) Quantum Software Testing in General; 2) Mutation Testing; 3) Search-Based Testing;

1) Quantum Software Testing in General: In [C01], the authors advocate for black-box testing on quantum computers to prevent a possible superposition collapse from white-box test-



Year	PP	TQ	RM	M
2018	1	0	0	0
2019	0	3	0	0
2020	3	3	1	1
2021	3	13	3	1
2022	2	13	2	1
2023	1	1	0	1
Total	10	33	6	4

Fig. 5: The number of papers by topic and year

TABLE I: Papers published in journals (10 papers)

Ref.	Title	Author(s)	Journal	Year	Topic
[P01]	ProjectQ: An open source software framework for quantum computing	Steiger *et al.*	Quantum	2018	PP
[P02]	On the definition of quantum programming modules	Sánchez and Alonso	Applied Sciences (Switzerland)	2021	TQ
[P03]	QuantumPath: A quantum software development platform	Hevia *et al.*	Software - Practice and Experience	2021	PP
[P04]	QRev: migrating quantum code towards hybrid information systems	Pérez-Castillo *et al.*	Software Quality Journal	2021	RM
[P05]	Software modernization to embrace quantum technology	Pérez-Castillo *et al.*	Advances in Engineering Software	2021	RM
[P06]	Design of classical-quantum systems with UML	Perez-Castillo and Piattini	Computing (Springer)	2022	M
[P07]	Studying efficacy of traditional software quality parameters in quantum software engineering	Faryal *et al.*	Optical and Quantum Electronics	2022	TQ
[P08]	Mutation Testing of Quantum Programs: A Case Study With Qiskit	Fortunato *et al.*	IEEE Transactions on Quantum Engineering	2022	TQ
[P09]	Comparing Quantum Software Development Kits for Introductory Level Education	Scekic and Yakaryilmaz	Baltic Journal of Modern Computing	2022	PP
[P10]	Formalization of Structural Test Cases Coverage Criteria for Quantum Software Testing	Kumar	International Journal of Theoretical Physics	2023	TQ

ing. They note that code reviews are feasible, but interactive debugging is challenging. In [C08], the authors propose various quantum program testing approaches. Functional testing requires multiple executions due to quantum program characteristics. White-box testing, particularly mutation testing, simulates potential programming errors. Model-based testing uses UML testing profiles for thorough circuit evaluation.

In [C02], the Scaffold QP language is enhanced with assertions and breakpoints, defining three breakpoint states: classical, superposition, and entangled. They use chi-square testing and contingency table analysis to compare program states with expected outcomes, aiding bug detection. The approach is validated with case studies like Shor's algorithm and Grover's search, showing its effectiveness in identifying incorrect states and potential bugs.

In [C28], a method for testing deterministic quantum algorithms directly on quantum machines through cost-effective quantum test case generation is proposed, aiming cost reduction with simulation performance. The algorithm defines qubits, creates output registers, runs the circuit under test, and compares outputs using Toffoli gates. If the outputs match, the desired output is obtained.

In [C14], the authors propose a five-step testing flow for validating Q# programs using QDK tools: 1) implement the

TABLE II: Papers published in conferences (43 papers)

Ref.	Title	Author(s)	Conference	Year	Topic
[C01]	On Testing Quantum Programs	Miranskyy and Zhang	ICSE	2019	TQ
[C02]	Statistical assertions for validating patterns and finding bugs in quantum programs	Huang and Martonosi	ISCA	2019	TQ
[C03]	Towards a Pattern Language for Quantum Algorithms	Leymann	QTOP	2019	M
[C04]	A tool for quantum software evolution	Jiménez-Navajas *et al.*	Q-SET	2020	RM
[C05]	Is Your quantum Program Bug-Free?	Miranskyy *et al.*	ICSE	2020	TQ
[C06]	Property-based Testing of Quantum Programs in Q#	Honarvar *et al.*	ICSEW	2020	TQ
[C07]	Quantum DevOps: Towards Reliable and Applicable NISQ Quantum Computing	Gheorghe-Pop *et al.*	GC Wkshps	2020	PP
[C08]	Quantum software testing	Usaola	QANSWER	2020	TQ
[C09]	The Quantum Software Lifecycle	Weder *et al.*	APEQS	2020	PP
[C10]	Towards a Quantum Software Modeling Language	Perez-Delgado and Perez-Gonzalez	ICSE	2020	M
[C11]	Quantum agile development framework	Hernandez Gonzalez and Paradela	QUATIC	2020	PP
[C12]	KDM to UML Model Transformation for Quantum Software Modernization	Jimenez-Navajas *et al.*	QUATIC	2021	RM
[C13]	Towards a Set of Metrics for Quantum Circuits Understandability	Cruz-Lemus *et al.*	QUATIC	2021	TQ
[C14]	Testing quantum programs using Q# and Microsoft quantum development kit	Mykhailova and Soeken	QSET	2021	TQ
[C15]	Bugs4Q: A Benchmark of Real Bugs for Quantum Programs	Zhao *et al.*	ASE	2021	TQ
[C16]	Identifying Bug Patterns in Quantum Programs	Zhao *et al.*	Q-SE	2021	TQ
[C17]	Modelling Quantum Circuits with UML	Perez-Castillo *et al.*	Q-SE	2021	M
[C18]	Muskit: A Mutation Analysis Tool for Quantum Software Testing	Mendiluze *et al.*	ACE	2021	TQ
[C19]	Non-functional requirements for quantum programs	Saraiva *et al.*	Q-SET	2021	TQ
[C20]	Poster: Fuzz Testing of Quantum Program	Wang *et al.*	ICST	2021	TQ
[C21]	QBugs: A Collection of Reproducible Bugs in Quantum Algorithms and a Supporting Infrastructure to Enable Controlled Quantum Software Testing and Debugging Experiments	Campos and Souto	Q-SE	2021	TQ
[C22]	QDiff: Differential Testing of Quantum Software Stacks	Wang *et al.*	ASE	2021	TQ
[C23]	Quantum Computing Platforms: Assessing the Impact on Quality Attributes and SDLC Activities	Sodhi and Kapur	ICSA	2021	PP
[C24]	Quito: a Coverage-Guided Test Generator for Quantum Programs	Wang *et al.*	ASE	2021	TQ
[C25]	Some Size and Structure Metrics for Quantum Software	Zhao	Q-SE	2021	TQ
[C26]	Generating Failing Test Suites for Quantum Programs With Search	Wang *et al.*	SSBSE	2021	TQ
[C27]	On Decision Support for Quantum Application Developers: Categorization, Comparison, and Analysis of Existing Technologies	Vietz *et al.*	ICCS	2021	PP
[C28]	Automatic Generation of Test Circuits for the Verification of Quantum Deterministic Algorithms	García de la Barrera Amo *et al.*	QP4SE	2022	TQ
[C29]	QMutPy: A mutation testing tool for Quantum algorithms and applications in Qiskit	Fortunato *et al.*	ISSTA	2022	TQ
[C30]	Mutation Testing of Quantum Programs Written in QISKit	Fortunato *et al.*	ICSE	2022	TQ
[C31]	Mutation-based test generation for quantum programs with multi-objective search	Wang *et al.*	GECCO	2022	TQ
[C32]	Using Quantum computers to speed up dynamic testing of software	Miranskyy	QP4SE	2022	TQ
[C33]	QuSBT: search-based testing of quantum programs	Wang *et al.*	ICSE	2022	TQ
[C34]	A Comprehensive Study of Bug Fixes in Quantum Programs	Luo *et al.*	SANER	2022	TQ
[C35]	A multi-lingual benchmark for property-based testing of quantum programs	Pontolillo and Mousavi	Q-SE	2022	TQ
[C36]	Metamorphic Testing of Oracle Quantum Programs	Abreu *et al.*	Q-SE	2022	TQ
[C37]	Asserting the correctness of Shor implementations using metamorphic testing	Costa *et al.*	QP4SE	2022	TQ
[C38]	Quokka: A Service Ecosystem for Workflow-Based Execution of Variational Quantum Algorithms	Beisel *et al.*	ICSOC	2022	PP
[C39]	A Tool For Debugging Quantum Circuits	Metwalli and Meter	QCE	2022	TQ
[C40]	Classical to Quantum Software Migration Journey Begins: A Conceptual Readiness Model	Akbar *et al.*	PROFES	2022	RM
[C41]	Silq2Qiskit - Developing a quantum language source-to-source translator	Hans and Groppe	CSSE	2022	RM
[C42]	Paraconsistent Transition Systems	Cruz *et al.*	EPTCS	2023	M
[C43]	Quantum Software Architecture Blueprints for the Cloud: Overview and Application to Peer-2-Peer Energy Trading	O'Meara *et al.*	SusTech	2023	PP

main quantum algorithm; 2) integrate it with the classical part; 3) simulate the quantum code; 4) estimate required resources; and 5) execute on quantum hardware. [C14] also introduces Quantum Katas, an open-source project for validating quantum algorithms with immediate feedback.

In [C24], the authors introduce Quito, a QP testing tool that generates test suites based on Input Coverage, Output Coverage, and Input-Output Coverage. Quito uses Wrong Output Oracle and Output Probability Oracle for correctness checks. Users provide input information, select coverage criteria, choose a significance level for OPO assessment, and define test generation budgets.

2) Mutation Testing: The mutation testing sub-category presents ten mutation testing papers, which consist of inducing faults along the generation of the mutants.

In [C29] [P08] [C30], the authors propose a new quantum mutation testing approach, introducing five quantum mutation operators and a tool called QMutPy. Meanwhile, the authors in [C31] explore classical mutation testing in QPs using a multi-objective, search-based approach. The approach generates minimal test cases to eliminate mutants across various difficulty levels. In [C18], the authors presented Muskit, a mutation analysis tool with a command line interface, GUI, and web application that can execute test cases on mutants and generate results for test analyses.

In [C06], the authors present QSharpCheck, offering a property specification language, architectural overview, and statistical analysis techniques. Their framework contributes a language for specifying test properties, a method for generating concrete test cases, and demonstrates effectiveness through mutation testing. [C35] provided a benchmark for property-based testing across Qiskit, Cirq, and Q#, evaluating frameworks by generating properties and mutants for each language. The study found Qiskit and Cirq excel in property-based testing, whereas Q# lacks generic testing libraries and optimization.

Quantum metamorphic testing involves defining metamorphic relations, which are rules that relate the program input to an output to determine its correctness. The authors in [C36] and [C37] define metamorphic relations specific to Shor's algorithm and introduce mutants to evaluate its effectiveness. In [C22], the authors introduce QDiff, a differential testing strategy for QP stacks. QDiff enhances input diversity through semantic-preserving gate transformations and mutations. Applied to Qiskit, Cirq, and Pyquil, QDiff generates logically equivalent QPs, optimizes them, and selects a subset for testing on noisy simulators or quantum hardware.

3) Search-Based Testing: The search-based testing subcategory uses search algorithms to find failing inputs and tests, accelerating defect detection in QPs.

In [C26] and [C33], QuSBT is introduced, using a genetic algorithm to generate test suites with maximum failing cases. [C32] uses Grover's search algorithm to identify error-causing inputs. [C20] proposes a search-based test input generator for unitary gate-based matrices and applies fuzz testing to detect buggy behaviors in quantum-sensitive branches.

Answer for the RQ1.1. Most testing papers used Qiskit or tools based on it, such as Muskit [C18] or Quito [C24]. Other tools include ProjectQ, Cirq, Pyquil, QCEngine, and Microsoft Quantum Development Kit. Out of the 21 papers analyzed, nine presented new tools. The most used testing strategies were mutation-based and search-based. Some QC testing approaches utilized quantum-specific elements to their advantage, while others adapted classical testing methods for QP. Both types of techniques impacted software testing, be it with the non-deterministic nature of QC or using its specific elements to ensure test coverage.

An interesting discussion here is regarding the conventional testing classification. A well-acceptable taxonomy of testing is composed of *testing phases* (unit, integration, system) and *testing criteria* (functional, structural, defect-based (mutation)). Observing the final set of testing papers it is not trivial to classify some works considering this taxonomy. Some authors explicitly classify their proposal in unit testing and mutation testing, but others do not.

RQ1.2 - How has QC impacted software quality?

To answer this RQ we considered twelve papers. In this topic RQ1.2, the works are papers linked to quantum software quality. This topic was divided in the following sub-topics: QP in General; Metrics; Bug Analysis;

4) QP in General: In [C19], the authors highlight crucial non-functional requirements for QP development, including optimizing for the maximum available qubits, circuit depth for stability, T gate limits, minimizing gates between unconnected qubits, and reducing unavailable gate usage on the target quantum device.

The authors in [P07] explores the impact of QC characteristics on software quality, comparing them with classical parameters based on ISO/IEC 25010 standards. The analysis covers eight standards given factors like the specialized environment, non-public availability, and the ability to accommodate big data, with Security being the most important. The paper highlights quantum computers' efficiency in cryptography, necessitating the development of quantum-resilient security algorithms.

5) Metrics: Three papers focus on defining terms and metrics to evaluate the quality of QP. In [P02], the authors introduce quantum programming through modularization, proposing metrics for module coupling and cohesion. In [C13], the authors present various metrics for assessing QPs, such as Qubit Cost, Gate Count, and Depth, aiming for a comprehensive source for analysis of QP nature. In [C25], the author focus on metrics for evaluating quantum algorithms, emphasizing code size, design size, and specification size, alongside structural metrics analyzing module complexity and interaction, providing insights into program quality.

The paper [P10] explores Cyclomatic complexity, which quantifies the number of independent paths and paths created due to the superposition of each control qubit. The method ensures that every path is tested at least once, ensuring coverage in testing QP programs. The paper also examines different coverage criteria based on single, two, and three qubits.

6) Bug Analysis: Several studies have proposed QP Testing benchmarks or bug collections to enhance QP. In [C21], the authors developed a benchmark to identify common QPs bugs and their corrections, addressing challenges like the lack of open-source projects and bug reproducibility. [C15] focuses on Qiskit, utilizing its control history to gather, replicate, and classify bugs. [C34] analyzes 96 real-world bugs, finding that most bugs are quantum-specific and often resolved with minor code adjustments. [C16] categorizes Qiskit bugs into initialization, gate operation, measurement, and deallocation issues, stressing the importance of safe programming practices. In [C05], the authors extend classical debugging tactics, such as backtracking, elimination, and brute force, to QC, addressing challenges like measuring variables in superposition and checking entanglement. In [C39], the author introduces the Quantum Circuit Slicer tool within a debugging framework, enabling vertical and horizontal division for sub-circuit generation and bug tracking.

Answer for the RQ1.2. The papers discussed QP, its metrics, and how to improve it while considering quantum-specific elements. Many papers have presented benchmarking tools such as Qbugs and Bugs4Q. These benchmarks use bugs from various quantum programming languages, for instance, Qiskit, Cirq, Q#, PyQuill, and ProjectQ. Those benchmarks identify, solve, or classify quantum bugs, improving the quality of QP. Similarly, a debugging tool was introduced and validated.

RQ2 - How has QC influenced software reengineering and modernization practices?

Research related to software reengineering/modernization in the realm of QC started around 2020 with the paper of Jiménez-Navajas *et al.*, [C04]. They present initial ideas regarding the problem of modernizing existing classical systems into hybrid systems and also how to integrate existing QP programs into hybrid systems. It also presents an adaptation of the canonical horseshoe model (reverse enginering, reestructuring and forward engineering) to cope with QP. They also raise some challenges of the quantum reengineering process.

This topic is the most related to knowledge extraction since every reverse engineering approach deals with recovering knowledge from legacy systems. In [C12], the authors introduce a model transformation approach from KDM (Knowledge Discovery Metamodel) to UML. This is a kind of transformation that can be used in the restructuring phase of the horseshoe modernization cycle. The transformations were developed in ATL (Atlas Transformation Language), being able to handle quantum entities like qubits and quantum gates. Quantum circuits are represented as activity diagrams using a quantum UML profile. The fact of using UML enables combining quantum algorithms with other classical software elements. Furthermore, this approach provides a graphical visualization of quantum algorithms with UML.

In [P05], the author advocates for a model-driven reengineering process to integrate quantum algorithms into existing classical systems. The focus is on the Reverse Engineering phase, which is divided into two modules: parsing Q# files to build an abstract syntax tree, and generating a KDM model. The goal is to ensure compatibility and scalability.

In [P04], it is presented QRev, a reverse engineering tool designed to generate quantum-adapted KDM models from Q# programs. The extended KDM model is able to represent classical and quantum information and their relationships. This technology-agnostic approach allows integrating QPs from different languages, a significant advancement in QP engineering. The KDM models provided by QRev can be used along with software modernization, facilitating refactorings and introducing new quantum functionalities.

Presented in [C41], Silq2Qiskit is a tool that automatically translates Silq basic quantum algorithms into Qiskit quantum circuits. The tool handles quantum function definitions, control flow, basic gates, and classical statements. The proposed library extends Qiskit with Silq's core concepts like Quantum Indexing and quantum data type emulation and demonstrates fully automatic and correct translations, successfully running Silq programs on IBM's quantum systems. The paper asserts the feasibility of a source-to-source translator to Qiskit and OpenQASM, indicating that a complete translation is possible.

Both classical and QP reengineering emphasize understanding existing systems, restructuring for efficiency and compatibility, and forward engineering for improvements. Specialized modeling techniques are required to handle quantum entities like qubits and quantum gates, as seen in [C12], which are unique challenges in quantum reengineering.

Additionally, as seen in [P05] and [P04], integrating quantum algorithms into classical systems poses integration and compatibility challenges. Tools like Silq2Qiskit, described in [C41], address these by enabling automatic translation between quantum languages, easing integration. These approaches align with modern software reengineering, where automation aids informed decisions on system updates.

Answer for the RQ2. The impact of QC on modernization and reengineering is substantial. One of the most significant impacts occurs when transitioning from a classical system to a hybrid one, which is likely to become the most common scenario. During the reverse engineering phase, it's crucial to identify quantum opportunities, elements of the system that could benefit from quantum reshaping. Subsequently, in the restructuring phase, there should be quantum-specific transformations that take classical software and produce a quantum version.

Quantum Reengineering can be understood as the process of adapting or leveraging algorithms or techniques from classical computing to QC. This includes adapting classical algorithms for use in quantum systems, using classical techniques to optimize quantum performance, or combining quantum and classical techniques for mutual benefits.

In conclusion, the main advances in software reengineering and modernization practices can be summarized as follows: the adaptation of classical algorithms in order to exploit the capabilities of QC, [C04]; integration between QPs, or from QPs into existing classical systems, [C41] and [P05], often facilitated by model-driven approaches; development of reverse engineering tools aimed at parsing existing classical codes and and generating representations suitable for quantum systems, [P04]; adoption of technology-agnostic approaches, [P04], which underscores the importance of interoperability and flexibility in the transition to QC; and the fostering of collaborative efforts to establish best practices for QP reengineering, [C40], guiding industry-wide adoption and integration into diverse application domains.

RQ3 - How to model quantum software?

This section explores different methodologies for modeling QP, defining a collection of five papers.

In [C03], the authors proposed the establishment of a pattern language for QC. This involves using structured documents with abstract descriptions of proven solutions to recurring problems, forming a network of related patterns. The objective is to create a pattern repository to store pattern documents and manage the connections between them, with the proposed PatternPedia repository serving this purpose.

In [C10], the authors introduce Q-UML, an extension of UML aimed at enabling software engineers to model QP details. They present two UML diagrams - class and sequence diagrams - to illustrate details of Shor's algorithm. A notable aspect of their proposal is the use of UML to represent details of a single algorithm, diverging from traditional UML usage focused on larger system representations. Modifications to the class diagram include bold font for quantum elements and double lines for quantum relationships. Similarly, in the sequence diagram, messages involving quantum information are depicted with double lines and in bold.

In [C17], a UML profile tailored for quantum circuit modeling is presented, containing stereotypes specific for representing quantum circuit, qubits, quantum logic gates, control qubits, measurements, and reset. This extension enhances UML's ability to depict domain-specific details and quantum circuits through diagrams. In contrast, [P06] proposes a UML profile for hybrid systems, expanding UML to cover structural and behavioral aspects of classical-quantum systems. It includes use case diagrams, class diagrams, sequence diagrams, activity diagrams, and deployment diagrams, facilitating the analysis and design of such systems.

In [C42], the authors introduced morphism for paraconsistent labeled transition systems (PLTS). After that, they define the category and algebra of PLTSs and their morphisms. The paper also presents the notions of simulation, bisimulation, and trace for PLTS. Circuits, which consider qubit decoherence as an error factor, are modeled using PLTS.

Answer for the RQ3.. This answer can be put in several ways and considering many views. The crucial point is that modeling QP is quite different from modeling conventional ones. Nowadays, QP is still confined to algorithms. It is difficult to find a QP that can be considered a ”system”. Therefore, most of the initiatives for modeling QP concentrate on algorithm-level details. As can be seen, most modeling efforts have focused on extending the Unified Modeling Language (UML) to include representations of quantum systems. This approach enables the retention of the classical systems standard modeling language accommodating quantum systems, thus facilitating the representation of hybrid systems. This approach allows using existing platforms for modeling purposes, such as Papyrus. The overall granularity focus of the papers is four high-granularity papers, analysis level, and two intermediate ones, project level.

RQ4 - How have development processes and platforms been shaped by QC?

This topic encompasses articles that delve into proposals on how QP affects software development processes and platforms. Beginning the development platforms subsection, [P03] introduces a platform for QP development grounded in principles of agnosticism and independence. The proposed software seeks to offer a comprehensive toolkit for QP development, including a circuit editor, annealer compositor, and code editor. The primary objective of the project is to deliver a toolkit capable of developing QP for any hardware or vendor.

Similarly, [P01], introduces ProjectQ, an open-source framework for developing QPs. This framework is a high-level quantum programming language built on Python, designed to be open and easy to learn for producing QP. ProjectQ features a modular architecture that supports customization of the compiler and back-ends, enabling writing code for different hardware platforms and adding new features such as simulators and circuit drawing.

In [C38], the authors provide an ecosystem that facilitates the execution of tasks, given a circuit in OpenQASM. Those tasks are: circuit execution service, error mitigation service, objective evaluation service and optimization service. Those services are provided through API requests.

In [C27], the authors present a categorization and taxonomy of tools, services, and techniques for QP development. They outline six main aspects: Quantum Cloud Services, Quantum Execution Resources, Transpilation and Compilation, Knowledge Reuse, Programming Language, and Classical-Quantum Integration. Additionally, the authors propose a conceptual framework for comparison to assist developers in selecting appropriate technologies.

In [P09], the authors focus on the pedagogical area, comparing 4 different quantum software development kits (QSDKs) based on how ready they are for newcomers. The authors evaluated Qiskit, Cirq, Forest (pyQuil) and ProjectQ in terms of installation, visualization and functionalities by executing 8 tasks proposed by Bronze, an introductory tutorial for quantum programming. The main contribution of the paper is exposing the strengths and weaknesses of each developing platform, providing a detailed benchmark of the platforms in various aspects in the development of QP.

[C23] proposes a paradigm shift in QP engineering across the software development life cycle. Enhancements include a comprehensive architecture for QC platforms, a refined programming model, and evaluation of architectural features' impact on quality attributes. The authors emphasize the importance of interoperability, maintainability, and verifiability unique to quantum platforms. They correlate performance requirements with software life cycle activities, identifying thirteen main characteristics of quantum projects and their impact on overall software quality.

In the process subclass, [C09] explores the lifecycle of QP, proposing a ten-phase methodology. They integrate classical SE principles with the current landscape of QP, including proprietary quantum hardware, Noisy Intermediate-Scale Quantum (NISQ) systems, and dependence on classical computing, to provide an overview of QP development.

[C11] explores agile methodologies in the realm of QC. While acknowledging that many characteristics of QP development are covered by agile methodologies, the authors note the need for adaptation to address specific quantum issues, like multidisciplinary teams and hybrid quantum-classical systems. The paper proposes a framework that combines elements from Scrum with traditional agile paradigms, such as incremental development and stakeholder collaboration, to better suit the needs of QP development.

In [C07], the authors introduce a DevOps methodology to address reliability issues in current NISQ computers. They propose a 5-step pipeline that regularly checks the state of various quantum computers and selects the most reliable one for calculations. The goal is to develop a methodology for producing reliable software on top of unreliable hardware.

Also in the DevOps theme, [C43] outlines a blueprint for a cloud service tailored to quantum development. It organizes key components in a DevOps pipeline and adapts classical practices for QC. Using a peer-to-peer energy trading application as a case study, the authors demonstrate the model and explore quantum advantages over classical methods with an Auction Matching Algorithm.

Answer for the RQ4. In summary, QC has spurred the development of specialized platforms to cater to the unique characteristics of QC: [P03]'s focus on a hardware-agnostic kit reflects a shift towards more flexible and adaptable quantum platforms; [P01]'s open-source framework puts emphasis on high-level programming languages, facilitating accessibility and ease of development; [C38]'s ecosystems streamline quantum algorithm's workflow, while [C09] and [C11] adapted methodologies acknowledges the need for adaptation in current life-cycle and agile methodologies to address the unique challenges and requirements of QP development. These developments represent a significant shift in the landscape of software engineering, focusing on increasing accessibility and hardware independence of QC, which reflects the growing importance and potential of QC technologies.

V. Conclusions and Research Opportunities

This paper presented a systematic mapping focusing on the impact quantum computing has on four software engineering topics. It is evident that the four topics/RQs do not cover several other aspects of the sinergy between QC and Software Engineering. However, we believe this can provide a good starting point for researchers identify research opportunities.

The final set of papers shows us a great interest of researchers in the testing area. This is not a surprise, since how to test quantum software emerges as a clear challenge. The main challenges are related to qubit superposition and the difficulty of testing quantum programs without affecting information in a way that causes a total collapse of the system.

Regarding the topic of software modernization, particularly within the phase of reverse engineering, an open research area pertains to the automatic detection of "quantum opportunities" within classical systems. The objective is to examine a model of a classical system and automatically pinpoint segments that could be potentially transformed into their quantum counterparts. To our knowledge, none of the existing literature has addressed this aspect.

It is interesting to observe that the KDM metamodel is used by some research groups in the context of systems modernization. Although KDM and UML were not originally designed to cope with quantum elements, they are extended (by profiles) to accommodate the quantum particularities. A very important detail is the agnostic nature of KDM, which is a platform and language independent model [13].

In contrast, there is a scarcity of papers related to Metrics/Terminology. Thus, the difficulty of standardizing and defining means to better analyze and understand the quantum part of programming becomes evident.

Although Quantum Computing is in its early stages, there is an undeniable collective effort to encourage its advancement and development. Consequently, a wide range of development platforms has emerged, each with its own characteristics, whether through quantum circuit simulation, commercially focused environments, open-source environments, or cloud-accessible development environments. The impact of Quantum Programming is evident.

Supplementary Material: All the material of this SM can be found here: https://zenodo.org/records/11094668

Acknowledgment

We would like to thank the financial support of Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES - Financial Code 001) and Programa de Educação Tutorial (PET)/ Fundo Nacional de Desenvolvimento da Educação (FNDE) / Brazillian Ministry of Education (MEC).

References

[1] N. Yanofsky and M. Manucci, *Quantum computing for computer scientists*. Cambridge University Press, 2008.

[2] IBM, "Introduction to quantum circuits," 2023.

[3] I. Sommerville, "Software engineering," Addison-Wesley Longman, Incorporated, 2004.

[4] P. E. Z. Junior and V. V. de Camargo, *A systematic mapping on quantum software development in the context of software engineering*, 2024. arXiv: 2106.00926 [cs.SE].

[5] E. F. Combarro, "Quantum computing foundations," in *Quantum Software Engineering*, M. A. Serrano, R. Pérez-Castillo, and M. Piattini, Eds. Cham: Springer International Publishing, 2022, pp. 1–24.

[6] M. Möller and C. Vuik, "On the impact of quantum computing technology on future developments in high-performance scientific computing," *Ethics and Information Technology*, vol. 19, no. 4, pp. 253–269, Dec. 2017.

[7] A. A. Clerk, K. W. Lehnert, P. Bertet, J. R. Petta, and Y. Nakamura, "Hybrid quantum systems with circuit quantum electrodynamics," *Nature Physics*, vol. 16, no. 3, pp. 257–267, Mar. 2020.

[8] J. Barzen, "From digital humanities to quantum humanities: Potentials and applications," in *Quantum Computing in the Arts and Humanities: An Introduction to Core Concepts, Theory and Applications*. Cham: Springer International Publishing, 2022, pp. 1–52.

[9] B. Sodhi and R. Kapur, "Quantum computing platforms: Assessing the impact on quality attributes and sdlc activities," in *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, 2021, pp. 80–91.

[10] M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Data encoding patterns for quantum computing," in *Proceedings of the 27th Conference on Pattern Languages of Programs*, ser. PLoP '20, Virtual Event: The Hillside Group, 2022.

[11] B. A. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep. EBSE 2007-001, Jul. 2007.

[12] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: A proposal and a discussion," *Requirements engineering*, vol. 11, Mar. 2006.

[13] B. M. Santos, A. de S. Landi, D. S. Santibáñez, R. S. Durelli, and V. V. de Camargo, "Evaluating the extension mechanisms of the knowledge discovery metamodel for aspect-oriented modernizations," *Journal of Systems and Software*, vol. 149, pp. 285–304, 2019.

Bus Ridesharing For Millions of Passengers

Xingyan Chen
SUN YAT-SEN UNIVERSITY
chenxy878@mail2.sysu.edu.cn

Zitong Chen
SUN YAT-SEN UNIVERSITY
chenzt53@mail.sysu.edu.cn

Abstract

In urban public transit, the ever-evolving and diversifying demands of the public pose significant constraints on bus operations due to their adherence to fixed routes. To meet the desire for low-cost and flexible travel, this paper proposes a high-capacity bus ridesharing service that can be implemented in any city worldwide. This service will transform buses from adhering to fixed routes to customizing personalized routes based on the destinations of ridesharing passengers. Passengers can submit their travel requests via mobile devices, and upon receiving these requests, the operator will dispatch suitable buses to provide service, ensuring that each passenger arrives at their destination within the allotted time. Utilizing urban public transit's passenger flow distribution, we have developed an experimental platform to emulate real-world bus travel behaviors. Our testing involved a genuine dataset of taxi requests and a simulated bus dataset representing the movements of 4.1 million individual in Chengdu city, China.

1. INTRODUCTION

As urban intelligence advances, the travel needs of the public become increasingly dynamic and personalized. To keep pace with this trend, we can innovate the bus operation model to better cater to the evolving demands of the public.

Ridesharing[6] is an intelligent matching service that utilizes information about passengers, drivers, and road networks to facilitate resource sharing. It not only reduces travel costs for passengers[1] and generates more income for drivers, but also piques the interest of both service providers and passengers[21]. Moreover, ridesharing plays a significant role in easing urban traffic congestion, representing a triple-win strategy for the advancement of smart city transportation. However, current carpooling systems, such as Uber, Lyft, DiDi, and Gaode, are primarily focused on the ride-hailing market, with limited capacity and high costs, making it difficult to meet the needs of the working class for large-scale, frequent, and low-cost travel. Although taking the bus is an economical option, the fixed operating mode of buses often results in longer travel times for passengers.

This paper, rooted in the practical context of public transit systems, conducts an in-depth exploration of the bus ridesharing issue and designs an innovative bus ridesharing service system to effectively tackle this challenge. The algorithm we propose eliminates the need for additional public funding as it can be seamlessly integrated into the existing bus infrastructure. Addressing this issue within the vast urban road network presents two major challenges: first, the passenger capacity of buses significantly exceeds that of taxis; second, each bus must simultaneously process multiple passengers' requests for origins and destinations, along with their time constraints, which essentially translates into the Vehicle Routing Problem with Time Windows (VRPTW)[16], a problem that has been proven to be NP-hard.

To the best of our knowledge, this study is the first to address ridesharing issues at the million-scale demand level. We generated passenger travel data based on the morning and evening rush hours of city public transportation commuting, as well as the average commuting distance. This data is used to simulate the entire bus travel process.

Our main contributions are as follows:

- This paper introduces an innovative bus ridesharing service that emulates the taxi model, offering personalized route planning to simplify the journey and ensure timely arrival for passengers.

- This paper presents efficient insertion and scheduling algorithms capable of swiftly managing real-time requests from a vast number of passengers.

- This paper develops a portable bus ridesharing platform that can be implemented in any city, utilizing urban road network and bus stop data from OpenStreetMap, without the need for additional public infrastructure investment.

DOI reference number: 10.18293/DMSVIVA2024-093

2. Related work

Ridesharing represents a quintessential carpooling service, with its research origins tracing back to the dial-a-ride problem introduced in 1975 [20][19]. This topic has been thoroughly investigated across various domains. Passengers can send their travel requests via smart mobile devices based on their work or life needs. After receiving a multitude of requests, vehicle operators record and analyze passenger demands to predict and forecast their needs [7][11][13], then proceed to dispatch orders[8][12] and plan vehicle routes[17]. Throughout this process, various constraints must be taken into account, such as waiting times, pricing, routes, dispatching costs, and passenger payments [3]. While taxi services offer a more personalized and efficient mode of travel, they come with a significantly higher cost of transportation.

The core concept of bus ridesharing is to offer a low-cost and mass-serving transportation solution. Compared to traditional ridesharing services that allow for flexible pick-up and drop-off at arbitrary locations with short waiting times, existing bus-ridesharing services often require passengers to make certain compromises regarding pick-up locations and waiting durations. For instance, [2]introduces a minibus planning algorithm designed to cater to pre-collected travel demands. This method first clusters travel requests with nearby origins and destinations and close request times, then plans routes for each group using a greedy approach. The algorithm necessitates a pre-existing database of all passengers' details, thus precluding the capability to offer services on a real-time basis. [4] proposes a greedy method for planning new bus lines based on taxi trajectories, employing DBSCAN to cluster the start and end points of taxi routes and using a greedy strategy to plan the new lines.Because the start and end points are random, some passengers may need to travel a long distance to reach their destination after clustering. [22]presents a high-capacity request matching and path planning algorithm based on clustering, which fundamentally merges multiple shortest path requests from various origins to destinations that are close to each other on the road network into a single trip. [15] discusses "Slugging," a variant of carpooling and hitchhiking where passenger A abandons their trip to join B's trip (i.e., A's trip is consolidated into B's). [14] formally defines the bus ride problem and proposes a large-scale bus ride service model where passengers can submit their travel demands through an online platform and are picked up once a sufficient number of people have gathered. In the proposed model, a better pick-up point is designed for passengers—selecting the location closest to both A and B, rather than having B wait for A.

Previous studies have often aggregated passengers across a broad area of multiple bus stops, necessitating that passengers congregate at specific locations and endure longer wait times to benefit from bus services. Our proposed algorithm diverges from this approach by significantly reducing passenger movement, permitting them to make bookings at the closest bus stops with the assurance of service within a 10-minute window. The most extensive dataset of passenger requests utilized thus far is the NYCTaxi dataset, comprising 400,088 authentic taxi trajectories from New York City on May 10, 2016. While these data stem from actual taxi operations, they fail to accurately mimic the commuting demands of public buses in metropolitan areas. To simulate a day's worth of public transit activity in Chengdu more authentically, we have carried out pertinent simulation research.

3. PRELIMINARIES

3.1. Basic definitions

Definition 1. ***Road Network****.A road network is represented by an undirected graph $G = (V, E)$, where graph G consists of a set of vertices V and a set of edges E . Each edge $(u, v) \in E$ in graph G is associated with a travel cost.*

We obtained city road network data from OpenStreetMap and the locations of city bus stops from Gaode, using the bus stops and road network data to form a new undirected graph G, where the set of vertices V now only includes bus stop locations, and the set of edges E represents the actual distances between two bus stops V. The number of bus stop locations in each city is significantly much smaller than the original road network vertices.Therefore We can record the shortest path distances between any two vertices $u, v \in V$ using a two-dimensional matrix, denoted as $dis(u, v)$.

Definition 2. ***Request*** *A request is denoted by $r =< o_r, d_r, p_r, s_r, e_r, k_r >$, where the origin $o_r \in V$, and the destination $d_r \in V$, with k_r representing the number of travelers. The request is posted on the Bus-Ridesharing (BRS) platform, referred to as the platform, at the posting time p_r, and must be picked up by a bus before the start time s_r to avoid undue waiting for passengers, the maximum waiting time for request is $s_r - p_r$. Additionally, the request must be delivered to the destination before the deadline e_r. To facilitate quick filtering, we record the stops near the origin and destination: Stops Near Original Location SNOL $= \{v_1, v_2, v_3, ..., v_n\} \in V$, and Stops Near Destination Location SNDL $= \{v_1, v_2, v_3, ..., v_n\} \in V$.*

Definition 3. ***Bus*** *A bus is represented by $B = \langle L, S, C \rangle$, where the bus's current location $L \in V$, S refers to the bus's trip schedule, and C denotes the bus's capacity. The capacity of a bus is the maximum number of passengers it* *can accommodate at any given time, and the current location L is the bus stop the vehicle is about to arrive at. We denote the set of all buses by $B = \{b_1, \dots, b_{|n|}\}$, where $|n|$ indicates the total number of buses.The vehicle speed in this paper is set according to the uniform speed of the vehicle type in the city.*

3.2. Trip schedule

The trip schedule of vehicle b_i is denoted by $b_i.S = \{l^0, l^1, l^2, ..., l^n\}$, which is a sequence of bus stops including the origin o_r or destination d_r of request r_i. Apart from l^0, which records the current location of the bus $b_i.L$ (that is, $l^0 = b_i.L$), we refer to the locations on the trip schedule as "stops" l^i. The path between any two adjacent stops l^{k-1} and l^k is called a "segment" denoted as (l^{k-1}, l^k).

Following prior research[18], we use three arrays, aar$[k]$, ddl$[k]$, and slk$[k]$, to record the expected arrival time, the latest arrival time, and the slack time for $b.S$ respectively:

The estimated arrival time arr$[k]$ records the estimated arrival time at stop l^k via the trip schedule.

The deadline time ddl$[k]$ records the latest acceptable arrival time at the stop l^k.

The slack time slk$[k]$ captures the maximum allowable additional travel time between segments (l^{k-1}, l^k).

For stop l^i, only the detour time ddl$[i]$ − arr$[i]$ is permitted to ensure its final arrival time ddl$[i]$. The detour between l^{k-1} and l^k will not only affect the arrival time at l^k but also the arrival time at all subsequent stops; hence, slk$[k] = \min\{\text{ddl}[i] - \text{arr}[i]\}$, for $i = k, \dots, m$. slk$[k]$ can be determined by referencing slk$[k+1]$, that is, slk$[k] = \min\{\text{ddl}[k] - \text{arr}[k], \text{slk}[k+1]\}$.

3.3. Candidate Bus Locations

To avoid long waiting times for requests, we can determine a distance dis$_{\text{original}}$ by multiplying the maximum waiting time $s_r - p_r$ by the vehicle speed. Any bus b_i with location $b_i.L$ within the distance dis$_{\text{original}}$ from o_r may potentially serve the request. We denote this set of buses as $B_{\text{candidate}} = \{b_1, \dots, b_n\}$. To quickly obtain $B_{\text{candidate}}$, we record the set of bus stops within the distance dis$_{\text{original}}$ from o_r, which we call SNOL. Similarly, we set up SNDL based on the same distance criteria, as detailed in the next section.

Example 1. *As shown in Fig.1, o_r, which is bus stop v_2, has a circle centered at v_2 with a radius of dis$_{wait}$. Station v_6 is outside the circle, so bus b_6 cannot reach v_2 by time s_r to service request r. Bus b_5 departs from station v_2 heading towards v_5. We assume the vehicle's current position is the next bus stop it will reach, so b_5 is currently at v_5. As it is outside the circle, it cannot service request r. Stops v_1, v_2, v_3, and v_4 are within the circle's range. We cannot retrieve v_4 even though the Euclidean distance between v_4 and v_2 is less than dis$_{wait}$. However, on the actual road network, the distance from v_4 to v_2 is greater than dis$_{wait}$, whether through v_5 , $v_3$or directly. Therefore, the SNOL for request r is $\{v_1, v_2, v_3\}$, and the candidate set of buses $B_{candidate}$ is $\{b_1, b_2, b_3\}$.*

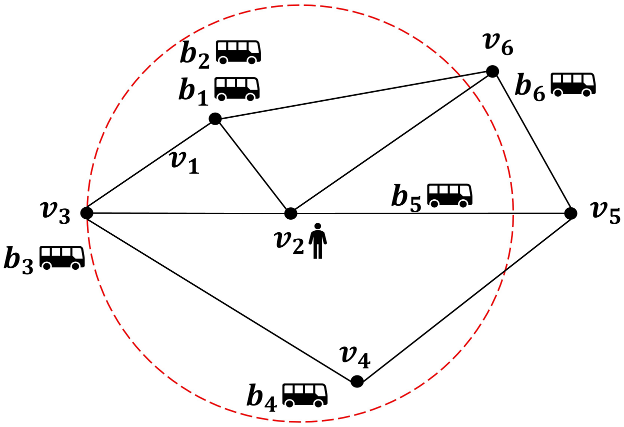


Figure 1: b_1,b_2 are at v_1, b_3 , b_4 are at v_3, b_5 leaves v_2 heading towards v_5, b_6 is at v_6, the radius of the circle is dis$_{\text{original}}$, and r is at v_2.

4. Bus-RideSharing FRAMEWORK

We employ the following simulation process: Record events occurring at each moment in seconds; at each moment, update the status of vehicles and passengers based on the events; process passenger orders at set time intervals, which involves matching passengers with vehicles. Our approach has three steps: **1.Select**: Facing a vast array of passengers and vehicles, we first narrow down the range of candidates; **2.Insert**: Based on the selection, determine whether vehicles and passengers can be paired and construct a bipartite graph between passengers and vehicles;**3.Matching**: Each edge in the bipartite graph represents a potential pairing relationship, followed by executing a maximum weight matching to allocate the matched passengers to the corresponding vehicles.

1.Select.For a new request r, we select each $b_i \in B_{\text{candidate}}$. If b_i is empty, r can be served by b_i. If $b_i.S$ is not empty and b_i's direction of travel aligns with r's requested direction, then r has a high probability of being served. SNDL is the set of stops around d_r. We check if $b_i.S$ will pass through SNDL. If it does, it indicates that the bus is traveling in roughly the same direction and might be able to detour to d_r to drop off the passengers. To determine this, we check if there is an intersection between $b_i.S$ and SNDL. However, due to the large scale, calculating the intersection for every bus and request would be

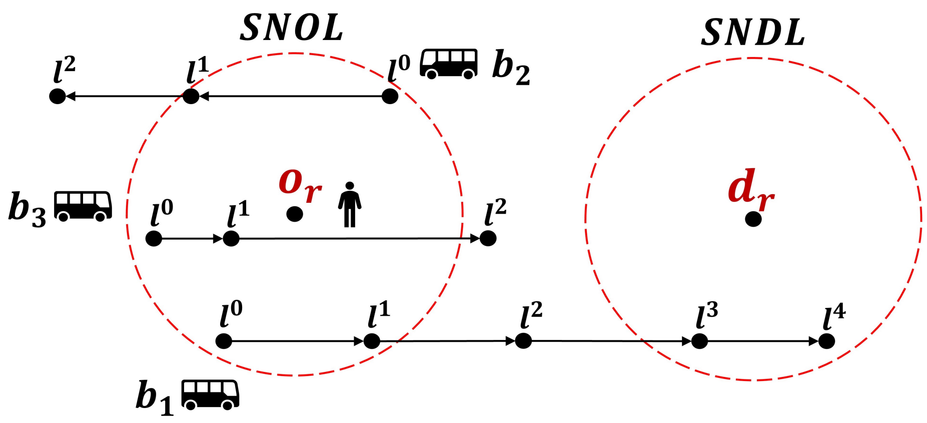


Figure 2: request and Candidate bus

time-consuming. Instead, we use a preliminary selection method: from $b_i.S = \{l^0, l^1, l^2, \ldots, l^n\}$, we select three points: l^0, $l^{n/2}$, l^n:

1) $(l^0 \in \text{SNDL}) \vee (l^{n/2} \in \text{SNDL}) \vee (l^n \in \text{SNDL})$

2) $\text{dis}(l^0, d_r) > \text{dis}(l^{n/2}, d_r) > \text{dis}(l^n, d_r)$

3) $b_i.S \cap \text{SNDL} \neq \emptyset$

b_i that satisfies 1) represents buses traveling in the same direction as r. For buses that do not satisfy 1) but satisfy 2), we then find the intersection between $b_i.S$ and SNDL, i.e., 3). If 3) is satisfied, then the bus is traveling in the same direction.

Example 2. *As shown in Fig.2, $b_1.l^4 \in$ SNDL, so b_1 satisfies condition 1), and request and b_1 are on the same route. b_2 is not on the same route because it does not satisfy conditions 1) and 2), as $dis(l^0, d_r) < dis(l^1, d_r) < dis(l^2, d_r)$. b_3 satisfies condition 2) but does not satisfy condition 3), so it is not on the same route.*

2.Insert.After filtering out the requests r and b that are on the same route, we find the optimal positions $\text{insert}(i, j)$ to insert o_r and d_r into $b.S$ under time constraints, minimizing the total distance of the updated $b.S$. Jaw et al.[10] proves that the complexity of this process is $O(n^3)$, where n is the length of $b.S$. We propose a new insertion algorithm:

$$\text{dis}_{\text{detour}} = \text{dis}(l^k, v_{\text{detour}}) + \text{dis}(l^{k+1}, v_{\text{detour}}) - \text{dis}(l^k, l^{k+1}) \quad (1)$$

Algo 1. *The detour distance is defined as:*(1)

The detour time $\text{time}_{\text{detour}}$ is obtained by dividing the detour distance $\text{dis}_{\text{detour}}$ by the bus speed. $b.C > k_r$ indicates that the bus can accommodate the requested number of passengers. We find the optimal boarding position in the set $b.S \cap \text{SNOL}$. If $\text{time}_{\text{detour}-o_r} < \text{slk}[i]$ and $\text{arr}[i-1] + \text{time}_{\text{detour}-o_r} < s_r$, it means the bus can pick up the passenger within the specified time. Then, we find the drop-off position in $b.S \cap \text{SNDL}$. If $\text{time}_{\text{detour}-o_r} + \text{time}_{\text{detour}-d_r} < \text{slk}[j]$ and $\text{arr}[j-1] + \text{time}_{\text{detour}-o_r} + \text{time}_{\text{detour}-d_r} < e_r$, it means the bus can drop off the passenger within the specified time. This reduces the time complexity to $O(m+n)$, where m is the size of $b.S \cap \text{SNDL}$ and n is the size of $b.S \cap \text{SNOL}$. As shown in Figure 2, o_r can be inserted between (l^0, l^1) and (l^1, l^2) of b_1, and d_r can be inserted between (l^2, l^3), (l^3, l^4), or at the end of l^4.

Algorithm 1 Insert in same direction

```
Require: $b$, $r$, SNOL, SNDL
Ensure: insert$(i, j)$
for $i \leftarrow 0$ to $n$ in $b.S \cap \text{SNOL}$ do
    if insert $o_r$ at $i$-th is feasible and $\text{dis}_{\text{detour-}o_r}$ is minimal then
        insert$(i) \leftarrow i$-th
    end if
end for
for $j \leftarrow 0$ to $n$ in $b.S \cap \text{SNDL}$ do
    if insert $d_r$ at $j$-th is feasible and $\text{dis}_{\text{detour-}o_r} + \text{dis}_{\text{detour-}d_r}$ is minimal then
        insert$(j) \leftarrow j$-th
    end if
end for
return insert$(i, j)$
```

3.Matching.Given a set of buses $B = \{b_1, b_2, \ldots, b_n\}$ and a set of requests $R = \{r_1, r_2, \ldots, r_m\}$. For any $b_i \in B$, it may be matched with multiple r, and the same applies to $r_i \in R$. B and R form a bipartite graph (as shown in Figure 3). We need to do matching in this bipartite graph. The KM algorithm can assign each b_i to the r_i with the minimal detour, but during peak travel times, the scale of B and R

becomes very large. The time complexity of the KM algorithm is $O((n+m)^3)$, which is impractical for real-time decision-making. We need to serve more R in a shorter time, so we use the HK algorithm. Although it cannot guarantee the optimal match for each b_i, its time complexity is $O(\sqrt{n}m)$, making it more suitable for large-scale problems.

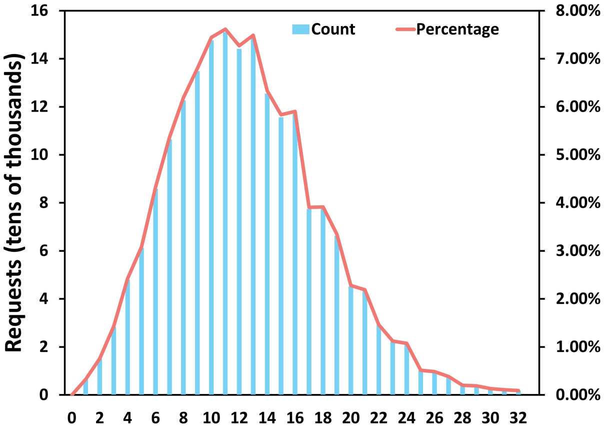
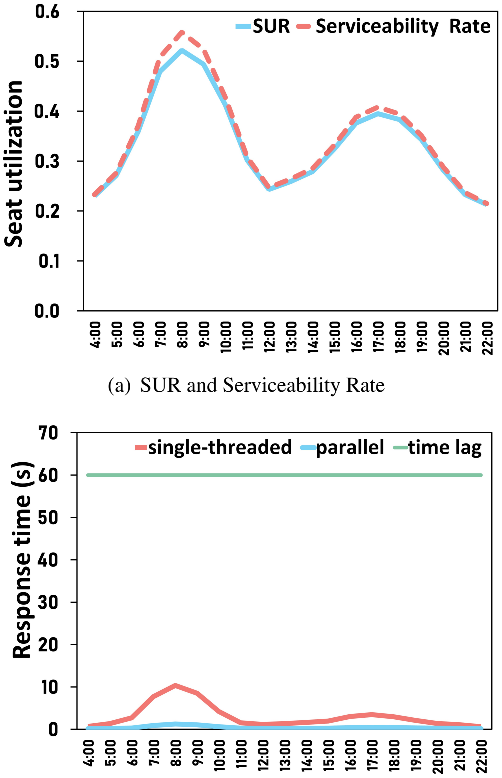


Figure 3: Distribution of distances

5. EXPERIMENTAL

Bus stops map. In this study, we use the vertex set V as the collection of bus stops in the city, and the edge set E is composed of any two adjacent bus stops. We downloaded the city's road network from OpenStreetMap, including the latitude and longitude of road vertices and the length of roads. Additionally, we crawled the latitude and longitude of the city's bus stops from Amap (Gaode Maps) and projected the bus stops onto the nearest roads.

Datasets.We conducted experimental evaluations on two datasets. The first dataset is a taxi dataset collected by DiDi Chuxing in Chengdu, China, and the second dataset simulates Chengdu's public bus trips. For the taxi dataset, we matched the pickup and drop-off locations of the requests to the nearest stations. For the bus dataset, we generated 2 million requests, with 4.1 million passengers, a bus capacity of 32, and a fleet of 14,000 buses. These values are based on actual figures provided by the Chengdu Public Transport Group [9].The $\text{dis}_{\text{original}}$ of SNOL is equal to the dis_{down} of SNDL. The origins o_r and destinations d_r are randomly generated on the map, and their distances follow a Poisson distribution as shown in Fig.3, with an average distance of 12 km. This data is sourced from the Chengdu Institute of Planning Design[5]. The distribution of requests p_r is simulated using a joint Gaussian distribution function, with 60% of the data derived from a Gaussian distribution representing morning trips with a mean of 8 AM, and 40% from a Gaussian distribution representing afternoon trips with a mean of 5 PM. The service time s_r is set to p_r plus 10 minutes. We set a relaxation parameter relax and define x as (2), relax relationship to s_r and e_r ,x is (3)



(a) SUR and Serviceability Rate

(b) decision-making time

Figure 4: Operation of the day

$$x = \frac{\text{distance from } o_r \text{ to } d_r}{\text{average vehicle speed}} \tag{2}$$

$$e_r = s_r + x \cdot \text{relax} \quad \text{relax} \in [1.5, 1.8] \tag{3}$$

If a request has not been serviced by s_r, it is considered abandoned. Both taxis and buses travel at constant speeds, with speeds of 48 km/h and 30 km/h, respectively.

Implementation.The experiments are conducted on a server with Intel(R) Core(TM) i7-12700F 2.10 GHz processor with 20 threads,running on the Linux operating system, and implemented in C++.

Table 1: Percentage of service

	taxi dataset	bus dataset
number of passenger	249,521	4,100,000
Served	234,036	3,812,217
percentage	93.4%	92.9%

Experiment Settings.The experiments of bus data set were repeated 20 times to take average values.The systems we developed were respectively designed for operating taxis (utilizing a taxi dataset with a vehicle capacity of 4 and a fleet size of 2000, representing the number of taxis in the dataset) and public buses (utilizing simulated datasets with a capacity set to the commonly used value and a fleet size representing the actual number of buses in Chengdu, which is 14,000) as shown in Table 1.

We recorded the bus operation throughout the day, as shown in Fig.4(a).**Seat Utilization Rate** $= \frac{\text{current number of passengers on board}}{\text{number of bus seats}}$, We recorded the average seat utilization rate of all buses at the current time.**Serviceability Rate** $= \frac{\text{current total number of passengers}}{\text{number of bus seats}}$, The current total number of passengers includes both those on board and those waiting, representing the proportion of current requests that can be accommodated by the buses. It can be seen that these two lines are very close to each other and change with the regular travel patterns of passengers, demonstrating that our algorithm can serve most requests.

We attempted decision-making at different time intervals, as shown in the Fig.5. When the time intervals were set to 30 seconds and 60 seconds, the number of successfully served passengers was almost the same, but the total simulation time for a day increased significantly. Although the time interval of 90 seconds required less time, its operational effectiveness was inferior to the former two. We compromised and decided to make decisions every 60 seconds, which also aligns better with real-world scenarios.

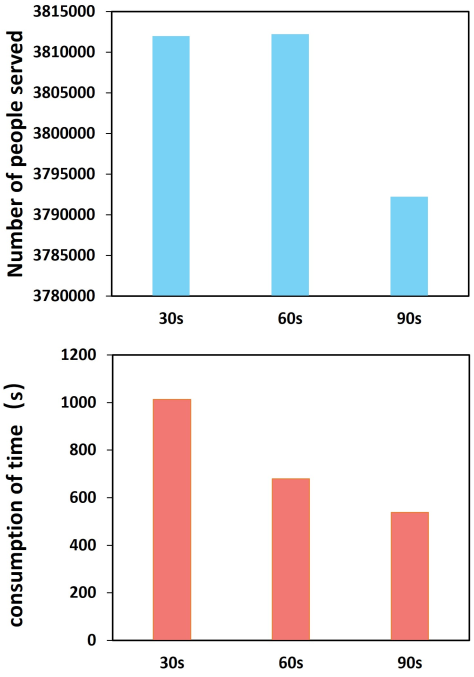


Figure 5: Results at different time intervals

Effective.The 93% service rate proves that our system is capable of carrying urban public transport.For the healthy operation of the system, it is essential to ensure that the time consumed by each decision-making process does not exceed the interval between decision-making processes. In Fig.4(b), we recorded the time required for each decision-making process throughout the day (shown by the red line) compared to the decision-making interval (shown by the green line). As we can see, the decision-making time is much shorter than the interval, even during peak hours. To handle potentially higher passenger volumes in the future, we used OpenMP and TBB parallel acceleration in the decision-making process, with the results shown by the blue line in the figure. By using passenger parallelism in selecting candidate vehicles for all passengers, the total processing time is significantly reduced. We can observe that even during peak hours (8 AM and 5 PM), the decision response time is not significantly affected by passenger volumes.According to the settings provided in [14], we compared the transportation of the same passenger volume by taxis and buses. With this setup, taxis successfully transported 3.8 million out of 4.1 million passengers. Taxis consume 9.12 XL/100KM,while buses consume 17.1 XL/100KM. To achieve the same passenger capacity, 32,000 taxis would be required. This means that using our algorithm could reduce the number of vehicles by 57% and fuel consumption by 32%. These findings indicate that bus pooling can efficiently utilize resources, demonstrating good energy efficiency and cost-effectiveness, thus alleviating urban traffic congestion and reducing carbon dioxide emissions.

6. CONCLUSION

In this paper, we first review and summarize the shortcomings of existing bus ridesharing services. Traditional bus ridesharing systems often face challenges such as inefficiency, slow response times, and poor user experience when matching passenger demands with bus schedules. To address these issues, we propose a novel real-time, large-scale bus ridesharing service. This service utilizes an intelligent matching algorithm to quickly pair passengers with buses traveling along similar routes, ensuring that pick-ups and drop-offs are completed within a constrained time frame, thereby enhancing overall service efficiency and reliability.

We also developed an innovative linear time insertion algorithm, which significantly reduces the computational cost of matching buses with passengers and optimizes the use of system resources. The core advantage of this algorithm lies in its ability to process large-scale data rapidly, enabling the system to make decisions in a very short period, thereby improving the response speed and efficiency of the entire bus ridesharing system.

To validate the effectiveness of our proposed solution, we conducted extensive simulations in a city-scale scenario involving millions of trips. The results demonstrate that our solution performs exceptionally well in these large-scale scenarios, proving its potential applicability to real world urban bus services. This not only indicates the practical value of our solution but also provides new theoretical insights and technical support for the development of smart cities. Our research offers important guidance for the future development of bus ridesharing services and presents a new perspective on optimizing urban transportation systems.

References

[1] Constantinos Antoniou, Dimitrios Efthymiou, and Emmanouil Chaniotakis. “Demand for emerging transportation systems: Modeling adoption, satisfaction, and mobility patterns”. In: (2019).

[2] Favyen Bastani et al. “A greener transportation mode: Flexible routes discovery from GPS trajectory data”. In: *19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2011, November 1-4, 2011, Chicago, IL, USA, Proceedings*. 2011.

[3] Lu Chen et al. “PTRider: A Price-and-Time-Aware Ridesharing System”. In: *Proceedings of the VLDB Endowment* 11.12 (2018), pp. 1938–1941.

[4] Seong Ping Chuah et al. “Bus routes design and optimization via taxi data analytics”. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 2417–2420.

[5] Chengdu Institute Of Planning Design. In: (). URL: `https://www.cdipd.org.cn/`.

[6] Erik Ferguson. “The rise and fall of the American carpool: 1970–1990”. In: *Transportation* 24.4 (1997), pp. 349–376.

[7] Yong Ge et al. “An energy-efficient mobile recommender system”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 899–908.

[8] Andrey Glaschenko et al. “Multi-agent real time scheduling system for taxi companies”. In: *8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary*. 2009, pp. 29–36.

[9] Chengdu Public Transport Group. In: (). URL: `https://www.cdgjbus.com/`.

[10] Jang-Jei Jaw et al. “A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows”. In: *Transportation Research Part B: Methodological* 20.3 (1986), pp. 243–257.

[11] Junghoon Lee, Inhye Shin, and Gyung-Leen Park. “Analysis of the passenger pick-up pattern for taxi location recommendation”. In: *2008 Fourth international conference on networked computing and advanced information management*. Vol. 1. IEEE. 2008, pp. 199–204.

[12] Junghoon Lee et al. “A telematics service system based on the Linux cluster”. In: *Computational Science–ICCS 2007: 7th International Conference, Beijing, China, May 27-30, 2007, Proceedings, Part IV 7*. Springer. 2007, pp. 660–667.

[13] Bin Li et al. “Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset”. In: *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE. 2011, pp. 63–68.

[14] Kaijun Liu, Jingwei Zhang, and Qing Yang. “Bus pooling: A large-scale bus ridesharing service”. In: *IEEE Access* 7 (2019), pp. 74248–74262.

[15] Shuo Ma and Ouri Wolfson. “Analysis and evaluation of the slugging form of ridesharing”. In: *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*. 2013, pp. 64–73.

[16] Martin WP Savelsbergh. “The vehicle routing problem with time windows: Minimizing route duration”. In: *ORSA journal on computing* 4.2 (1992), pp. 146–154.

[17] TANG Xinmin WANG Yuting HAN Songchen. “Aircraft taxi route planning for A-SMGCS based on discrete event dynamic system modeling”. In: *Second International Conference on Computer Modeling and Simulation2010*, pp. 167–173.

[18] Yongxin Tong et al. “A unified approach to route planning for shared mobility”. In: *Proceedings of the VLDB Endowment* 11.11 (2018), p. 1633.

[19] Nigel Henry Moir Wilson, Richard Wayne Weissberg, and John Hauser. *Advanced dial-a-ride algorithms research project*. Tech. rep. 1976.

[20] Nigel HM Wilson et al. *Advanced dial-a-ride algorithms*. Tech. rep. 1975.

[21] Wei Zhang et al. “Research on taxi pricing model and optimization for carpooling detour problem”. In: *Journal of Advanced Transportation* 2019 (2019).

[22] Haojia Zuo et al. “High-capacity ride-sharing via shortest path clustering on large road networks”. In: *The Journal of Supercomputing* 77.4 (2021), pp. 4081–4106.

AI-augmented Automation for Real Driving Prediction: an Industrial Use Case

Romina Eramo
Dept. of Communication Science
University of Teramo, Italy
reramo@unite.it

Hamzeh Eyal Salman
Dept. of Software Engineering
IT Faculty, Mutah University, Jordan
hamzehmu@mutah.edu.jo

Matteo Spezialetti
Dept. of Inf. Eng., Comp. Science and Math.
University of L'Aquila, Italy
matteo.spezialetti@univaq.it

Darko Stern
Dept. of Research Program Management
AVL, Austria
darko.stern@avl.com

Pierre Quinton
Dept. of Methodology and R&D
AVL, Austria
pierre.quinton@avl.com

Antonio Cicchetti
Dept. of IDT
Mälardalen University, Sweden
antonio.cicchetti@mdu.se

***Abstract*—The risen complexity of automotive systems requires new development strategies and methods to master the upcoming challenges. Traditional methods need thus to be changed by an increased level of automation, and a faster continuous improvement cycle. In this context, current vehicle performance tests represent a very time-consuming and expensive task due to the need to perform the tests in real driving conditions. As a consequence, agile/iterative processes like DevOps are largely hindered by the necessity of triggering frequent tests.**

This paper reports on a practical experience of developing an AI-augmented solution - based on Machine Learning and Model-based Engineering - to support continuous vehicle development and testing. In particular, historical data collected in real driving conditions is leveraged to synthesize a high-fidelity driving simulator and hence enable performance tests in virtual environments. Based on this practical experience, this paper also proposes a conceptual framework to support predictions based on real driving behavior.

***Index Terms*—Continuous software engineering, DevOps, Machine Learning, Automotive, Real Driving Emission testing.**

I. Introduction

Software plays an increasingly important role in modern vehicles. With the introduction of new features, new traffic regulations, system and security updates, automotive software requires the continuous verification and validation of new software versions, even after production. Thus, the traditional software life cycle, with slow feedback and manual interaction, is being replaced with a faster automatic feedback cycle and rapid continuous improvements.

With the advent of DevOps principles [18], [24], the (automotive) system engineering would benefit from supporting a continuous development involving a smooth continuum of information from design to runtime, and vice versa. Moreover, many leading companies have started to apply Artificial Intelligence (AI) principles and techniques for IT operations (AIOps) [12], [16], to rethink the DevOps pipeline through continuous monitoring, alerting, and remediation securely and reliably.

Models have a central role in vehicle development. Physical simulation models have been developed and optimized during the last decades and have reasonable maturity [32]. Moreover, software-driven vehicle components are subject to constant further development, for instance, novel models to meet the forthcoming emission standards are needed. As a consequence, new methods to support full continuous software and system engineering processes are required. The ongoing European *AIDOaRt* project[1] notably intends to address such issues. The project aims at providing a model-based framework to more efficiently support the continuous development of modern systems via AI-augmentation [8], [20].

This paper reports on a practical experience of developing an AI-augmented solution - based on Machine Learning (ML) - to support continuous vehicle development and testing. The experience is based on a real industrial use case headed by AVL[2], one of the industrial partners of the AIDOaRt project. In particular, new challenges have been raised with the introduction of more stringent emissions legislation. In fact, the *Real Driving Emissions* (RDE) test procedures [21] have been introduced in the EU aiming to evaluate nitrogen oxides (NOx) and particulate number (PN) emissions from passenger cars during on-road operation. In addition, more recent RDE legislation (euro7 compliance) extends the existing RDE criteria towards a much wider scope. In this context, AVL aims at improving the significance of test results including their evaluation in different vehicle development stages as well as the accuracy of simulation models. In fact, RDE test procedures tend to be time-consuming and expensive in real environments, and considering the needs for continuous development and testing they represent a bottle-neck in the process.

In this paper, we present a conceptual approach that, starting from the modeling of the driver's behaviour, defines the core components of the solution for the prediction of RDE in

DOI reference number: 10.18293/DMSVIVA2024-107

[1] AIDOaRt ECSEL-JU project: https://www.aidoart.eu/

[2] AVL List GmbH, Graz (Austria) http://www.avl.com/

virtual environments. The solution is based on model-based engineering (MBE) and ML: models are used to represent the necessary concepts that are included in the prediction; ML is used to generate high-fidelity simulations of the driver behaviour, and hence to enable the evaluation of driving emissions.

This paper is structured as follows. In section II, we provide the necessary background to understand our proposal. Section III presents the real driving prediction framework. An application of this framework is provided in section IV. Experimental results are discussed and evaluated in section V. Finally, the paper is concluded in section VIII.

II. Background

This section describes the basic concepts and context of the scope of the developed framework.

A. Basic concepts

Model Driven Engineering (MDE): MDE allows raising the level of abstraction and thus improving the ability to engineer and handle complex systems [35]. The use of models as purposeful abstractions of systems and environments is also increasing within the industry (e.g., digital twining [5]). While first-generation MDE tools mainly focused on generating code from high-level models, they now also address model-based testing, verification, measurement, tool/language interoperability, or software evolution, among many other software engineering challenges. In system and software engineering, MDE contributes by 1) providing better abstraction principles and techniques (e.g., for the handled data), 2) facilitating the automation of engineering activities, and 3) supporting technology integration among all the covered design and development activities.

Artificial Intelligence and Machine Learning (AI/ML): The dissemination of Artificial Intelligence (AI), including Machine Learning (ML), principles and techniques in a regulated industry enables systems to decide and act in a more and more automated manner: it is used by companies to exploit the information they collect to improve the products and/or services they offer [2]. Lately, AI/ML is also impacting all aspects of the system and software development lifecycle, from specification to design, testing, deployment, and maintenance, with the main goal of helping engineers produce systems and software faster and with better quality while being able to handle ever more complex systems and software [9], [22], [37].

B. The AIDOaRt approach

Fig. 1 provides a conceptual overview of the global solution based on the AIDOaRt project: it highlights the key principles and concepts that should be considered as the foundation of this work.

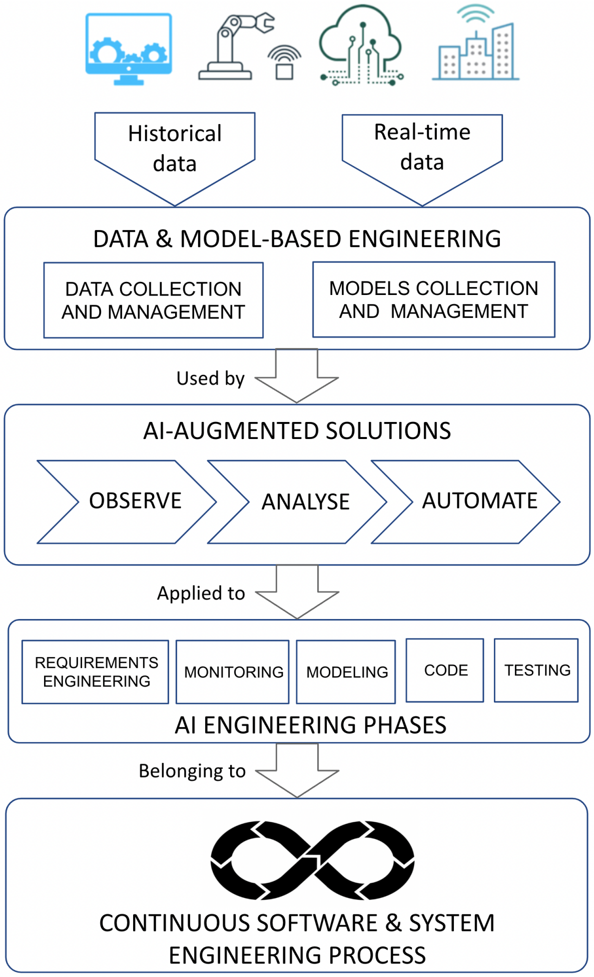


Fig. 1: AIDOaRt approach

The overall component consumes different kinds of data, including runtime data (e.g., IT monitoring, log events, etc.) and design data produced during the software development process (e.g., software models, design documentation, traceability information, source code, etc.). All the collected data and models will be processed and stored; the *Data & Model-based Engineering* component is intended to support the standard DevOps practices by providing methods and tools for the data and models collection and management. The *AI-augmented Solutions* component aims to enhance DevOps tool-chains (cf. existing DevOps tools [11]) by employing AI and ML techniques in multiple *Engineering Phases* of the system development process (e.g., requirements, monitoring, modeling, coding, testing, etc.). In an AIOps-enabled context, AI-augmented tools should support: 1) the monitoring of runtime data (such as logs, events, and metrics [36]) and software data and their traceability (namely *Observe*); 2) the analysis of both historical and real-time data (namely *Analyze*); and 3) the automation of development and operation activities (namely *Automate*). These capabilities will consume available design-time and runtime data that, according to MBE principles, should be made available to stakeholders as design-time and runtime models, respectively.

The aim is to extend existing techniques and introduce novel solutions, enhancing the state of the art of, for instance, requirements engineering, monitoring, and testing, that already includes mechanisms supporting/leveraging data analysis [7], [27], [38]. Moreover, search-based techniques have been investigated to automate MBE-related activities such as language engineering, model transformation, and model versioning [6]. Nonetheless, especially when dealing with mission-critical systems, the automated generation of artifacts raises verification and validation issues, e.g., for certification purposes [25].

After the data acquisition and management/preprocessing, the AI-augmented solutions may provide different support to the DevOps pipelines. In this work, we propose a model-based and AI-augmented solution for predicting human driver behavior (addressing the component *analysis* of the Figure) in the AVL RDE use case, described in detail in the next section.

C. The Real Driving Emissions case

AVL is a large independent company that deals with the development, simulation, and testing of power-train systems and their integration into the vehicle. New challenges have been raised with the introduction of more stringent emissions legislation. Particularly, procedures [21] for RDE testing have been introduced in the EU to evaluate nitrogen oxide (NOx) and particulate number (PN) emissions from passenger cars. Comparing the vehicle performance against the prescribed regulations in real driving conditions is very time-consuming, subject to unforeseen testing conditions, and very expensive. Therefore, AVL is developing a high-fidelity driving simulator to reproduce the RDE test procedure in a virtual environment. The current AVL's Smart Mobile Solution - Route Studio[3] is composed of a set of models for the route, human driver, and vehicle (see Figure 2). The most critical component of the Route Studio Simulator is the human driver model since it needs to reproduce the behavior of a human driver as accurately as possible on the selected route for testing and with arbitrarily generated traffic conditions.

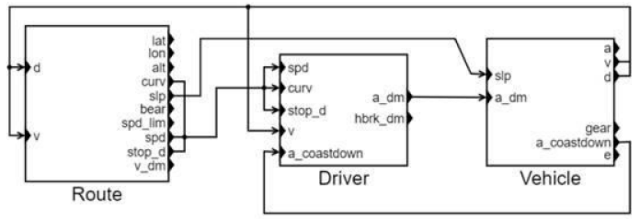


Fig. 2: AVL Route Studio Simulator architecture.

In the current implementation, the behavior of the human driver is based on a rule-based parametric model, that due to modeling simplification lacks the fidelity required for high-fidelity RDE estimations. The deviations between a human driver and a rule-based parametric model mostly come from the fact that human drivers have more complex behaviors that are hard to encapsulate with a heuristic model. Figure 3 show an example of a speed profile for both real driving (in blue) and a simulated one (in yellow) for a selected route.

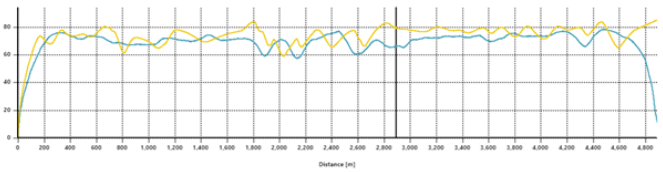


Fig. 3: Example of speed profile for a selected route (real driving in blue, simulated speed in yellow)

In the AIDOaRt project collaboration, we aim to improve the fidelity of the human driver behaviour model by developing a data-driven model which simulates human-like driving on any arbitrary test route and under various traffic conditions. Such ML models will lead to a better estimate of engine exhaust emissions, and thereby reduce the burden and cost of assessing vehicle compliance with strict emissions legislation.

III. Real Driving Prediction framework

Figure 4 presents a conceptual framework described using the MODA framework [14], a conceptual modeling [32] framework that aims at supporting the description of data-centric systems in terms of models, data, and transformations. By proposing a conceptual framework, we do not discuss particular tools or technologies, but we categorize the different roles and the relationships of artefacts on a conceptual level. In particular, the figure shows a simplified class diagram for real driving prediction; it aims at designing components, classes, and their relationships for the prediction of driver behaviour. The objective of this framework is to enable predictions through the analysis of historical data and updated data. It is composed of two main parts: i) the spatial model and predictors configuration (part I of Fig. 4) and ii) the domain-specific classes and relations describing the driving behaviour model we considered to use the framework (part II of Fig. 4) [26].

A. Model

This package includes one or more models of the system under analysis or its parts (represented by the class *Model*). In particular, it contains models that reflect the system and its environment in a descriptive manner, representing current or past aspects of the actual system, facilitating understanding, and enabling analysis [19].

B. Prediction

This package contains the abstract class *Predictor* that encapsulates a (predictive) model used to predict information that has not been measured, allowing decision-making and trade-off analyses. This can include models for analysis, simulation, and ML (e.g., different types of ML algorithms like Random Forest, Ordinary Least Square, KNN, etc.).

[3] https://www.avl.com/documents/10138/6781105/SMS_Simulation+Package_Solution+Sheet.pdf

C. Data

This package contains different features and metrics definitions. These are *Feature*, *VolatileFeature*, and *Metric*.

D. Driving behaviour

This package represents driving behaviours and related elements that determine such behaviours. The main elements are the environment, the vehicle, and the driver. The *DrivingModel* class extends the class *Model* and consists of at least one *Driver*, one *Vehicle*, and one *Route*. In particular:

- The settings for a specific driver are put in the *Driver* class. Driver's personal data such as *reaction speed*, *target velocity*, *action*, and *driving style*, will affect the driving behaviours. Driving style refers to *normal driving*, *zigzag driving*, *risky acceleration*, or *risky lane changing*. *Reaction speed* and *action attributes* refer to reactions toward other vehicles on the route. The *action* attribute takes the following attributes: *accelerating*, *decelerating*, and *maintaining* the current speed.
- To describe vehicles in detail, we introduce the *Vehicle* class. It contains attributes that influence driving behaviours, including *vehicle size*, *weight, engine type, acceleration, and average emissions*. Note that the *engineType* attribute is an enumeration literal, e.g. petrol, or hybrid.
- The *Route* class is the entry point for configuring the environment for the driving scenario. It includes settings such as the *speed limit*, *slope*, *curvature*, *weather conditions* and *road conditions*. The weather condition attribute takes the following values: *slipperiness* and *wind* while the road condition attribute has the following values: *highway, urban, mountain*. Each route is composed of an *HereMap* instance that allows customers to access a database of locations and map information. Moreover, a route is composed of signalization details that can be represented by *traffic signs* and *traffic lights*, *road shapes*.

This package represents a simplified version of a complete metamodel for defining driver behavior. However, it aims to be generic enough to suit many applications. In the next section, an application example is shown.

IV. Application to the RDE case

In this section, we present the application of the proposed framework on the RDE case, described in Sect. II-C. In particular, according to the RDE case providers, we focused on a simple case whose goal is to estimate the average behavior of drivers based on route data (see Route class in the framework). Thus, our goal is not to analyze individual driver behavior by considering, for instance, onboard sensors (see Driver and Vehicle classes in the framework) but rather to be able to estimate the average speed of drivers on a specific stretch of road, as the result affects emissions.

A. Data Acquisition

Even if our experiment involved a subset of the dataset, we initially acquired all the data provided by AVL through AIDOaRT project (see Table I). The data was collected from special equipment placed on a car. Data refers to several drivers and different driving paths: highways, mountains, etc. The data collected is real driving recordings in time series format (time-based data on *vehicle speed, throttle/brake pedals, curvature, road gradient, GPS coordinates, etc.*). All of these data are continuous 1D signals with a sample rate of 1Hz up to 10Hz. The recorded data are then stored as a matrix of the form $M \times N$ where M is equal to the number of measured channels (features) and N is equal to the number of recorder values.

B. Features Selection

To reduce the unnecessary computational cost, we filter out all the channels (features) that are not relevant for modeling human driver behavior. This filtering process is performed by the usecase provider based on its importance for driver behavior modeling. To that end, we take a close look at the channels (features) in the dataset (see Table I) and found that there are three types of channels: ① channels (resp. their corresponding data) are collected by two-way (snapshots and real driving recordings), ② channels (volatile features like distance) are calculated from other features (speed and time), and ③ metric channels (such as speed). The following channels are considered to train the data model while others are discarded: *spd_lim(16), tfc_flw(17), traf_lig(18), tfc_sgn(19), toll_booth(22), curvature(23), and slope(26)*. The *velocity_kmh_raw(9)* feature is considered as a target (output). Numbers in parentheses after each channel refer to channel ID in Table I.

The discarded features are excluded for different reasons. Firstly, some features are derived or computed from other features. For example, distance can be computed using time and speed. Secondly, some features are repeated but are computed in different ways. For example, *d, d_integrated_raw, d_raw,* represent the distance feature. Thirdly, some features have a negligible impact on the prediction, such as *alt_raw, lat_raw, lon_raw, sat_raw*.

C. Pre-processing

When we explored the features' data in the dataset, we found that each feature's data has a different scale. For example,*velocity_kmh_raw* feature ranges from 0 to 144 while values of *tfc_sgn* are integers ranging from 0 to 42. For this reason, we standardized each feature, except *velocity_kmh_raw*, before proceeding with the training. The standard score (Z_i) of a value (X_i) is calculated as follows:

$$Z_i = \frac{X_i - mean(X)}{Std\_Dev(X)} \qquad (1)$$

where *Mean* and *Std_Dev* are the mean and the standard deviation of feature X, respectively.

The *velocity_kmh_raw* feature represents the target feature or class. We decided to address the speed inference by posing the task as a classification problem. Therefore, since it ranges from 0 to 144 Km/h, we discretized it by dividing each data

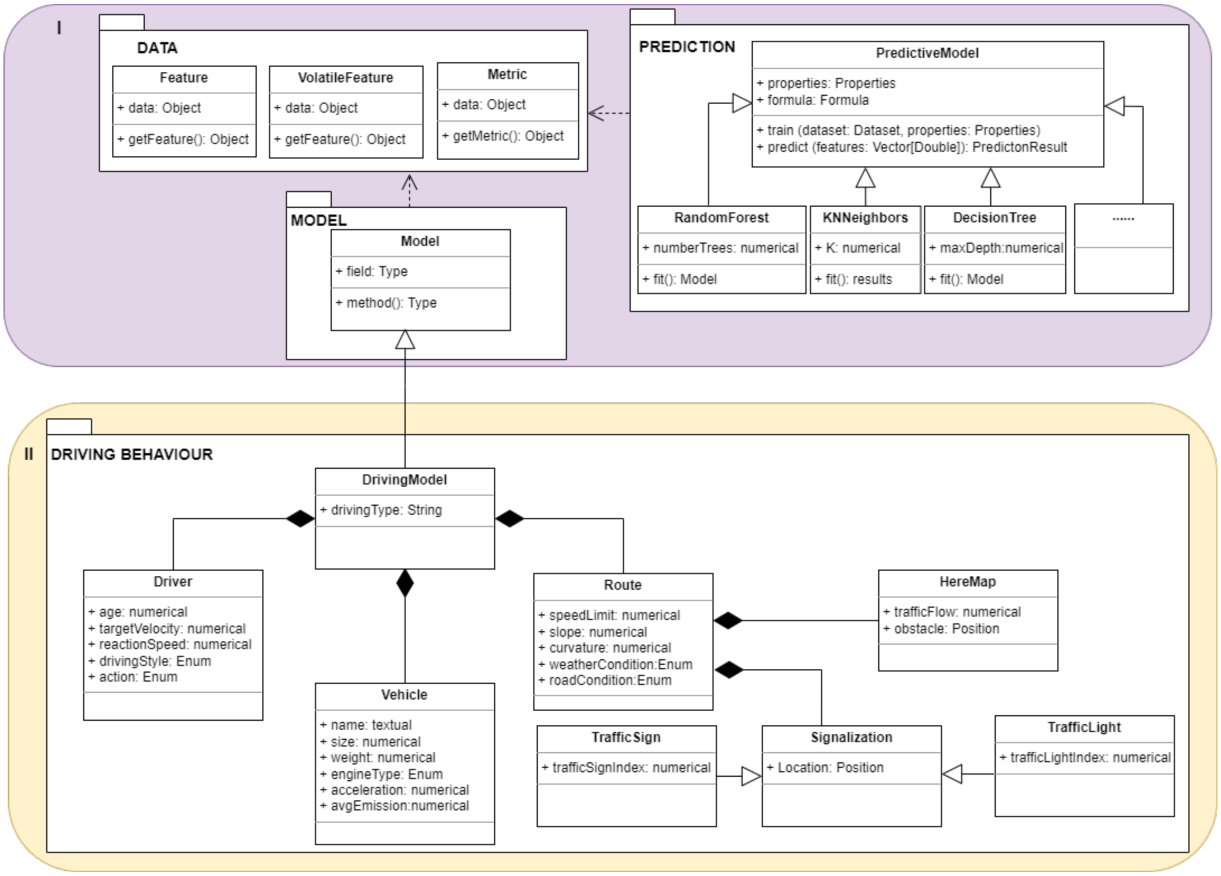


Fig. 4: Conceptual Framework for Real Driving Prediction

point X_i by 10 and rounding down the result to the closest integer. In this way, for example, any speed in the interval $[0, 10)$ has been mapped into class 0, any speed in the interval $[10, 20)$ into class 1, and so on.

D. Driving Behaviour Prediction

In the literature, there are many different ML algorithms for classification problems and there is no certain ML algorithm fit to all datasets. Choosing an ML algorithm depends on the type and size of the dataset of interest.

In this research work, we compare the performance of mostly widely used ML algorithms using Python scikit-learn packages to determine which algorithm is the best fit for the collected dataset. These algorithms are *GradientBoosting, DecisionTree, RandomForest, LogisticRegression, KNNeighbors, GaussianNB, LinearSVM, and AdaBoost*. The target algorithm should be able to predict the driver behaviour on the basis of the real dataset available and selected features on any arbitrary test route.

V. Experimental Results and Evaluation

A. Evaluation Procedure and Metrics

The ML algorithms used in this research work are applied to the dataset, which is split into 90% for training and 10% for testing. For classification purposes, the splitting process is performed with a random shuffle, and we deal with the dataset not as time series data, accordingly. Also, the training data is split further into five portions (cv=5) in a process called cross-validation (CV for short). The following procedures are followed for each portion:

1- ML model is trained using CV-1 portions.
2- The generated model is validated using the remaining portion.

The following standard evaluation metrics for classification solutions are used to evaluate the performance of these algorithms. The values of these metrics are scaled between 0 and 100%. Our aim is to find an ML algorithm that maximizes these values. In these metrics, *TP, TN, FP, FN* refers to *true positive*, *true negative*, *false positive*, and *false negative* prediction for each class, respectively.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5}$$

TABLE I: Raw Real Driving Features

ID	Feature	Description
1	velocity_raw	Raw imported vehicle speed
2	t	Time
3	speed_raw	Engine speed imported
4	alt_raw	Altitude imported
5	lat_raw	Latitude imported
6	lon_raw	Longitude imported
7	sat_raw	Nbr of satellite used for the measurement
8	d_integrated_raw	Distance calculated out of the imported vehicle speed integration
9 *	**velocity_kmh_raw**	Raw imported vehicle speed
10	d_raw	Distance calculated out of the latitude and the longitude
11	lat	Snapped latitude
12	lon	Snapped longitude
13	alt	Altitude from Here Maps
14	d	Distance out of snapped longitude and latitude
15	here_slope	Slope from Here Maps
16*	**spd_lim**	**Speed limit from regulation**
17*	**tfc_flw**	**Average speed from Here Maps**
18*	**traf_lig**	**Traffic light index (until 6) to indicate the number of traffic lights**
19*	**tfc_sgn**	**Traffic sign index**
20	sgn_loc	Localisation of the traffic sign (1=Left, 2=right, 3=above)
21	conf	Confidence value from the snapping
22*	**toll_booth**	**Index for toll booth**
23*	**curvature**	**Road curvature in 1/m**
24	curvature_rad	Road curvature in rad
25	bearing	Yaw of the vehicle
26*	**slope**	**Slope calculated from the Here Maps altitude**
27	alt_corr	Corrected altitude

* Selected features for training the prediction model

We adjusted the hyperparameters of these algorithms to be the default values in the Python scikit-learn API. We used these default parameters without deliberate adjustment to ensure a fair comparison in the experiments among the ML algorithms. Also, these default values are often chosen by the library developers to work well in a wide range of scenarios.

B. Results Discussion

The dataset includes 27 features. Since some features have a minor impact on the target prediction, we selected the features listed in Table I. The selected data was used for the train prediction model of several popular ML algorithms (i.e., *GradientBoosting, DecisionTree, RandomForest, LogisticRegression, KNNeighbors, GaussianNB, LinearSVM, and AdaBoost*).

Table II reports the performance results of the prediction models generated by the considered classifiers in the validation stage. As shown in this table, the prediction capability of *RandomForest* and *DecisionTree* classifiers is the best over other classifiers. The performance measures for these classifiers are [Precision: %90 - %91, Recall: %90 - %91, F1-score: %90 - %91, Accuracy: %90 - %91]. The KNN classifier is excluded from the validation stage. This is because there is no generated model by KNN classifier to validate. The KNN classifier depends on the majority rule for k nearest neighbours to classify unseen data [3].

Table III lists performance results of prediction models built by ML algorithms of interest in terms of Precision, Recall, and F1-score. These algorithms were applied to entire dataset records. *DecisionTree, RandomForest, and KNNeighbors* were the three top models among other classifiers where all metric values were greater than 90% (for DecisionTree and RandomForest) and over 80% for KNNeighbors. The highest metric values were produced by RandomForest as its values were 92%. The worst metric values were produced by *LogisticRegression, GaussianNB, LinearSVM, and AdaBoost* where their metric values were unable to achieve 50%. The metric values of the remaining classifiers (*GradientBoosting*) take a range between them.

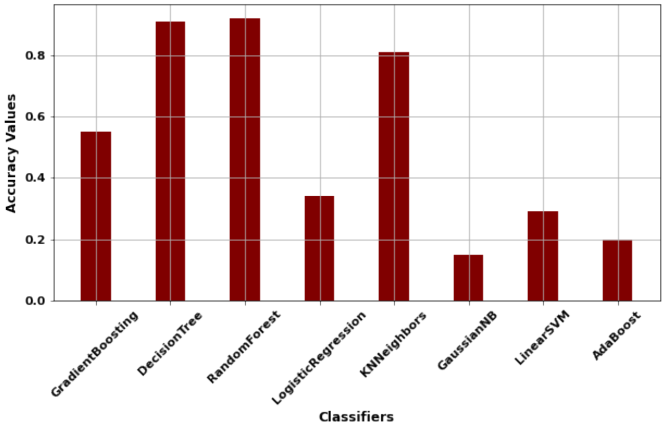


Fig. 5: Performance results of several classifiers.

Figure 5 shows the comparison results among the classifiers of interest in terms of accuracy metric. Similar to what is noted in Table III, the figure shows the three top prediction models are built using *DecisionTree, RandomForest, and KNNeighbors*. The highest accuracy results are produced by *RandomForest* classifier. Also, the worst accuracy values were produced by *LogisticRegression, GaussianNB, LinearSVM, and AdaBoost*.

Tables IV shows detailed results of the RandomForest (RF) classifier, which is the best fit on the considered dataset. We worked on a multi-class classification problem; in fact, the dataset contains 15 classes, and each class represents a speed range (see preprocessing subsection IV-C). Although there is a large number of classes in the dataset, RF achieves high metrics values (Precision, Recall, F1-score, and Accuracy) for each class. RF provides consistent and encouraging results, as demonstrated by the accuracy values of 92%.

C. Observations and Lessons Learned

We identified the following points as threats to the validity:

- Each driver's behavior differs from the other. For example, there are normal driving, aggressive driving, drowsy driving, etc. Our prediction model is built using only normal driving data as normal driving is the usual case.
- In the current approach, we neglect the non-deterministic nature of environmental conditions like traffic conditions, due to the fact that it is not recorded when collecting data of the real driving cycle. In future work we plan

TABLE II: Cross validation results with CV=5 for all considered classifiers.

	GradientBoosting		**DecisionTree**		**RandomForest**		**LogisticRegression**		**GaussianNB**		**LinearSVM**		**AdaBoost**	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
Weighted Precision	0.56	0.0	0.90	0.0	0.91	0.0	0.28	0.01	0.19	0.02	0.22	0.01	0.17	0.03
Weighted Recall	0.55	0.0	0.90	0.0	0.91	0.0	0.34	0.0	0.19	0.03	0.29	0.0	0.19	0.03
Weighted F1-Score	0.54	0.0	0.90	0.0	0.91	0.0	0.26	0.0	0.10	0.03	0.19	0.0	0.13	0.02
Weighted Accuracy	0.55	0.0	0.90	0.0	0.91	0.0	0.34	0.0	0.19	0.03	0.29	0.0	0.19	0.03

TABLE III: Obtained results using all considered classifiers.

	GradientBoosting	**DecisionTree**	**RandomForest**	**LogisticRegression**	**KNNeighbors**	**GaussianNB**	**LinearSVM**	**AdaBoost**
Weighted AVG Precision	**0.56**	**0.91**	**0.92**	**0.25**	**0.81**	**0.18**	**0.20**	**0.18**
Weighted AVG Recall	**0.55**	**0.91**	**0.92**	**0.34**	**0.81**	**0.15**	**0.29**	**0.20**
Weighted AVG F1-Score	**0.54**	**0.91**	**0.92**	**0.26**	**0.81**	**0.07**	**0.19**	**0.15**

TABLE IV: Evaluation results of the *RandomForest* classifier

Class	**Precision**	**Recall**	**F1-score**	**Support**
0	0.99	0.99	0.99	8134
1	0.94	0.93	0.93	2498
2	0.91	0.90	0.91	4556
3	0.90	0.90	0.90	5984
4	0.88	0.90	0.89	7053
5	0.88	0.87	0.88	5339
6	0.92	0.92	0.92	3786
7	0.93	0.92	0.92	3072
8	0.95	0.93	0.94	2283
9	0.93	0.93	0.93	1511
10	0.94	0.96	0.95	1646
11	0.84	0.81	0.83	499
12	0.78	0.79	0.79	285
13	0.88	0.82	0.85	222
14	0.64	0.47	0.54	15
Accuracy			**0.92**	**46883**
Weighted AVG	**0.92**	**0.92**	**0.92**	**46883**

to explore generative models such as variational autoencoders (VAE) or generative adversarial networks (GANs) to model environmental conditions.

VI. Related Work

In this section, we present the most recent and relevant research works to this study. We categorize these works into three categories based on the type of algorithm used to predict the driver behaviour [29]: *rule-based algorithms, ML-based algorithms, and digital twins.*

A. Rule-based Driver Behavior prediction

Rule-based algorithms are also called threshold-based algorithms. They are a set of algorithms that depend on the predefined threshold for monitored variables or factors to assign driver behavior to some class. In [33], Radoslav proposed an approach based on thresholds computed using data collected from smart mobile sensors to detect the following deriving events: acceleration, deceleration, left turn, right turn, lane change to left, lane change to right. In [30], Murphey et al. presented an approach based on the number of aggressive maneuvers to classify the driver behavior into: calm below 50%, aggressive above 100%, and normal otherwise. Another classification for driver behavior is proposed by [15] and [28]. This classification is based on fuel consumption or, in general, energy consumption.

The main limitation of all previous-mentioned works is that they depend on a single parameter and therefore the robustness and accuracy of the results are considerably limited. Another research direction relied on fuzzy logic to manage multiple parameters for driver behavior prediction. The research works in this direction are also based on predefined thresholds but are able to include more parameters whilst keeping its simplicity, robustness, easy understanding and low computational order [17] [23] [34] [39]. However, these works are also limited in terms of the number of variables and data that can process. Also, they depend on thresholds computed using expert intervention.

B. ML-based Driving Behavior Prediction

A variety of ML-based driving behavior prediction approaches (with different purposes) are proposed. They employed different types of learning to achieve this prediction purpose, such as supervised learning, unsupervised learning, and combined unsupervised and supervised learning [29] [4]. In this section, we focus only on supervised learning approaches as they are the closest to our work presented in this article.

The supervised learning approaches depend mainly on using driving data as training data to build a prediction model.

In [31], Campoverde et al. proposed an approach to estimate emissions by applying ML to an important set of OBD data. The main aim of this approach is to determine the selected gear by the driver as the emission can be estimated based on the gear number used. This is achieved by integrating ANN, k-means clustering, and random forest algorithm. In [13], Chen et al. conducted an empirical study to evaluate the performance of different ML algorithms to classify the behavior of specific drivers, on the base of the sensor technology installed on the car. Although their purpose is different from ours (in fact, the authors want to recognize a specific driver based on the data collected), the results are interesting. They reported that there is no single machine-learning algorithm that fits all problems. Based on their obtained data, their experiments showed that the random forest approach is a good fit for identifying driving behaviors. In [10], Karginova et al. made a comparison between multiple ML algorithms for the purpose of classifying the driver's driving style. These algorithms are KNN, NN, decision tree, and random forest. The KNN achieved the best performance within the nearest neighbor's inherent limits when clustering for K = 4 or 5.

C. Digital Twins

In [1], Dygalo et al. proposed to produce a digital twin of active vehicle safety systems for a proper simulation and correct system design. The digital twin consists of a set of modules ranked according to their priority. For example, "Wheel" and "Vehicle Body" modules were given top priority as they define the general parameters of the vehicle's movement and location in physical space. This way of ranking modules within the system helps to detect inconsistencies in the top-priority modules at the earliest stages, thus tracking any errors and promptly correcting them.

VII. Acknowledgment

We would like to thank Mutah University for funding the registration of this conference.

VIII. Conclusion and Future Work

This paper reported on a practical experience of developing an AI-augmented solution, based on Machine Learning and Model-based Engineering, to support continuous vehicle development and testing. We presented how historical data, collected in real driving conditions, is leveraged to synthesize a high-fidelity driving simulator and hence enable performance tests in virtual environments. Based on this practical experience, this paper also proposed a conceptual framework to support predictions based on real driving behavior.

In future work, we aim to extend the Driving Behaviour package employing a complete metamodel describing the domain. Moreover, we aim to propose a Digital Twin framework [19]; in automotive, a digital twin of the product comprises the entire car, its software, mechanics, electrics, and physical behavior. This allows for simulation and validation of each step of the development to identify problems and possible failures before producing real parts. We aim to focus on modeling the behavior of considered physical systems to make predictions by using several ML techniques. Also, we aim to validate the framework through several use cases in different domains (including the automotive and RDE) by using different implementations.

References

[1] Principles of application of virtual and physical simulation technology in production of digital twin of active vehicle safety systems. *Transportation Research Procedia*, 50:121–129, 2020. XIV International Conference on Organization and Traffic Safety Management in Large Cities (OTS-2020).

[2] *Gartner Predicts the Future of AI Technologies*, accessed: 28.07.2022. https://www.gartner.com/smarterwithgartner/gartner-predicts-the-future-of-ai-technologies/.

[3] H. A. Abu Alfeilat, A. B. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, H. S. Eyal Salman, and V. S. Prasath. Effects of distance measure choice on k-nearest neighbor classifier performance: A review. *Big Data*, 7(4):221–248, 2019. PMID: 31411491.

[4] M. Aliramezani, C. R. Koch, and M. Shahbakhti. Modeling, diagnostics, optimization, and control of internal combustion engines via modern machine learning techniques: A review and future directions. *Progress in Energy and Combustion Science*, 88:100967, 2022.

[5] F. Bordeleau, B. Combemale, R. Eramo, M. van den Brand, and M. Wimmer. Towards model-driven digital twin engineering: Current opportunities and future challenges. In *ICSMM 2020*, pages 43–54. Springer, 2020.

[6] I. Boussaïd, P. Siarry, and M. Ahmed-Nacer. A survey on search-based model-driven engineering. *Automated Software Engineering*, 24:233–294, 2017.

[7] L. C. Briand. Novel applications of machine learning in software testing. In *QSIC '08*, pages 3–10, 2008.

[8] H. Bruneliere, V. Muttillo, R. Eramo, L. Berardinelli, A. Gómez, A. Bagnato, A. Sadovykh, and A. Cicchetti. Aidoart: Ai-augmented automation for devops, a model-based framework for continuous development in cyber–physical systems. *Microprocessors and Microsystems*, 94:104672, 2022.

[9] L. Burgueño, M. Kessentini, M. Wimmer, and S. Zschaler. MDE intelligence 2021: 3^{rd} workshop on artificial intelligence and model-driven engineering. In *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS 2021 Companion, Fukuoka, Japan, October 10-15, 2021*, pages 148–149. IEEE, 2021.

[10] S. Byttner, N. Karginova, and M. Svensson. Data-driven methods for classification of driving styles in buses. In *SAE 2012 World Congress & Exhibition*. SAE International, apr 2012.

[11] N. Ceresani. The periodic table of devops tools v.2 is here, June 2016. https://blog.xebialabs.com/2016/06/14/periodic-table-devops-tools-v-2/, last accessed on 30.06.2021.

[12] S. G. Charley Rich, Pankaj Prasad. Market guide for aiops platforms, id g00378587. Technical report, Gartner Research, 2019.

[13] W.-H. Chen, Y.-C. Lin, and W.-H. Chen. Comparisons of machine learning algorithms for driving behavior recognition using in-vehicle can bus data. In *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 268–273, 2019.

[14] B. Combemale, J. Kienzle, G. Mussbacher, H. Ali, D. Amyot, M. Bagherzadeh, E. Batot, N. Bencomo, B. Benni, J. Bruel, J. Cabot, B. C. Cheng, P. Collet, G. Engels, R. Heinrich, J. Jezequel, A. Koziolek, S. Mosser, R. Reussner, H. Sahraoui, R. Saini, J. Sallou, S. Stinckwich, E. Syriani, and M. Wimmer. A hitchhiker's guide to model-driven engineering for data-centric systems. *IEEE Software*, 38(4):71–84, 2021.

[15] A. Corti, C. Ongini, M. Tanelli, and S. M. Savaresi. Quantitative driving style estimation for energy-oriented applications in road vehicles. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3710–3715, 2013.

[16] Y. Dang, Q. Lin, and P. Huang. Aiops: Real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 4–5, 2019.

[17] D. Dörr, D. Grabengiesser, and F. Gauterin. Online driving style recognition using fuzzy logic. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1021–1026, 2014.

[18] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. Devops. *IEEE Software*, 33(3):94–100, 2016.

[19] R. Eramo, F. Bordeleau, B. Combemale, M. van den Brand, M. Wimmer, and A. Wortmann. Conceptualizing digital twins. *IEEE Softw.*, 39(2):39–46, 2022.

[20] R. Eramo, V. Muttillo, L. Berardinelli, H. Bruneliére, A. Gómez, A. Bagnato, A. Sadovykh, and A. Cicchetti. Aidoart: Ai-augmented automation for devops, a model-based framework for continuous development in cyber-physical systems. In F. Leporati, S. Vitabile, and A. Skavhaug, editors, *24th Euromicro Conference on Digital System Design, DSD 2021, Palermo, Spain, September 1-3, 2021*, pages 303–310. IEEE, 2021.

[21] European Commission. Commission Regulation (EU) 2018/1832 of 5 November 2018 amending Directive 2007/46/EC of the European Parliament and of the Council, Commission Regulation (EC) No 692/2008 and Commission Regulation (EU) 2017/1151. https://eur-lex.europa.eu/eli/reg/2018/1832/oj, 2019.

[22] M. Felderer, E. P. Enoiu, and S. Tahvili. Artificial intelligence techniques in system testing. In *Optimising the Software Development Process with Artificial Intelligence*, pages 221–240. Springer, 2023.

[23] E. Gilman, A. Keskinarkaus, S. Tamminen, S. Pirttikangas, J. Röning, and J. Riekki. Personalised assistance for fuel-efficient driving. *Transportation Research Part C: Emerging Technologies*, 58:681–705, 2015. Technologies to support green driving.

[24] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer. What is devops? a systematic mapping study on definitions and practices. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, XP '16 Workshops, New York, NY, USA, 2016. Association for Computing Machinery.

[25] F. Leofante, N. Narodytska, L. Pulina, and A. Tacchella. Automated verification of neural networks: Advances, challenges and perspectives. In press, 2018.

[26] G. Lyan, J.-M. Jézéquel, D. Gross-Amblard, and B. Combemale. DataTime: a Framework to smoothly Integrate Past, Present and Future into Models. In *MODELS 2021 - ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems*, pages 1–11, 2021.

[27] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe. Toward data-driven requirements engineering. *IEEE Software*, 33(1):48–54, 2016.

[28] V. Manzoni, A. Corti, P. De Luca, and S. M. Savaresi. Driving style estimation via inertial measurements. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 777–782, 2010.

[29] C. Marina Martinez, M. Heucke, F.-Y. Wang, B. Gao, and D. Cao. Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):666–676, 2018.

[30] Y. L. Murphey, R. Milton, and L. Kiliaris. Driver's style classification using jerk analysis. In *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, pages 23–28, 2009.

[31] N. D. Rivera-Campoverde, J. L. Muñoz-Sanz, and B. d. V. Arenas-Ramirez. Estimation of pollutant emissions in real driving conditions based on data from obd and machine learning. *Sensors*, 21(19), 2021.

[32] D. Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *Computer*, 39(2):25–31, 2006.

[33] R. Stoichkov. *Android smartphone application for driving style recognition*. PhD thesis, Technische Univ, 2013.

[34] F. U. Syed, D. Filev, and H. Ying. Fuzzy rule-based driver advisory system for fuel economy improvement in a hybrid electric vehicle. *NAFIPS 2007 - 2007 Annual Meeting of the North American Fuzzy Information Processing Society*, pages 178–183, 2007.

[35] H. Thompson, M. Reimann, and D. Ramos-Hernandez. *Platforms4CPS, Key Outcomes and Recommendations*. Steinbeis-Edition, 2018.

[36] G. Valente, T. Fanni, C. Sau, T. Di Mascio, L. Pomante, and F. Palumbo. A composable monitoring system for heterogeneous embedded platforms. *ACM Transactions on Embedded Computing Systems*, 2021.

[37] Z. Wan, X. Xia, D. Lo, and G. C. Murphy. How does machine learning change software development practices? *IEEE Transactions on Software Engineering*, 47(9):1857–1871, 2021.

[38] S. Wolny, A. Mazak, M. Wimmer, and C. Huemer. Model-driven runtime state identification. In H. C. Mayr, S. Rinderle-Ma, and S. Strecker, editors, *EMISA 2019*, volume P-304 of *LNI*, pages 29–44. Gesellschaft für Informatik e.V., 2019.

[39] J.-S. Won. *Intelligent energy management agent for a parallel hybrid vehicle*. PhD thesis, Texas A&M University, 2003.

Automatic API Upgrade for jQuery Library

Ning Li, Yuhan Chen, Liangyu Chen
East China Normal University
Shanghai, China
lychen@sei.ecnu.edu.cn

***Abstract*—Third-party libraries play a significant role in software development, offering great assistance to programmers. However, they also increase the maintenance costs of projects. Upgrading these libraries can be time-consuming and laborious, particularly when dealing with deprecated APIs. jQuery is a widely used third-party library in JavaScript programs, and currently, there is no existing tool that can automatically upgrade its version. To address this upgrade problem of third-party libraries, this paper proposes an approach for automatically upgrading the version of jQuery. Firstly, official API documents are obtained as the basis for migration rules. Then, the abstract syntax tree is used to parse javascript code, and a replacement algorithm is designed to replace obsolete APIs, thereby achieving automatic upgrading of jQuery versions in the project. Finally, we selected 10 highly popular GitHub projects that included jQuery dependencies to conduct experiments and do verifications of jQuery API version upgrades. Compared to the @type version prompt plugin provided by VSCode[1] , our migration tool achieves** 74.2% **accuracy while VSCode gets** 61.4%**. This demonstrates our tool has a better recognition scope and accuracy. Our tool can automatically identify and replace obsolete API invocations with ones supported in the new version, effectively upgrading jQuery in the project code and enhancing the software quality.**

***Index Terms*—API Replacement, JavaScript Library, jQuery, Abstract Syntax Tree**

I. Introduction

Third-party libraries play a crucial role in software development by providing a wealth of features and solutions. Utilizing these libraries avoids writing code from scratch, thereby increasing development efficiency. Additionally, these libraries are optimized through version iterations to enhance user experience and offer ongoing maintenance support.

jQuery is the most well-known JavaScript (JS) library, widely favored by frontend developers for its HTML element manipulation, event handling, and Ajax requests. It holds a 94.4% market share in early 2024 [1]. However, jQuery has undergone continuous API modifications and structural changes through version iterations, such as function renaming, parameter changes, and even deprecations. These API changes during updates are often troublesome [2]: deprecated functions post-update can prevent projects from achieving previous functionalities or even running correctly, which is the reason that developers are reluctant to upgrade with the latest versions [3]. Unfortunately, there is currently no integrated tool to handle API changes during jQuery version upgrades automatically. Therefore, it is very important to enable developers to upgrade outdated jQuery dependencies in their projects, free from the limitations of older jQuery functionalities, and address deprecated API issues during upgrades.

In this paper, we propose an automated tool to solve the problem of jQuery version migration. The jQuery migration tool has the following features. First, it visually displays all deprecated APIs between jQuery versions. The tool collects all changing API information from the jQuery API website using Python crawlers and forms a document about API differences, assisting developers in handling API changes during jQuery version upgrades. Second, it automatically detects deprecated or unusable API locations in the project after version changes and automatically modifies the code through a replacement algorithm based on an abstract syntax tree (AST) to fit the new API specification. Finally, the effectiveness of the jQuery migration tool is validated through experiments on several actual GitHub projects.

In summary, the main contributions of this paper are as follows:

- We parse jQuery official documents to obtain API usage and version records, identify deprecated APIs, and get corresponding new ones.
- We propose a migration method for jQuery API, realize a jQuery API replacement method based on API documents information idea, with the help of abstract syntax tree to realize the API mapping, change the idea is also applicable to other library API migration. Although this migration method cannot directly migrate to other third-party libraries, the method is a trial for upgrading API versions of other third-party libraries, and this approach applies to all other third-party libraries.
- We implement a jQuery API migration tool to automatically upgrade old dependencies in real projects. The tool is verified on 10 projects from Github that were selected and achieved an average accuracy of 74.2%. Compared with VsCode's deprecated function tool, our tool has a better recognition range and accuracy rate for jQuery API deprecation. The tool code and datasets in this paper can be accessed at https://github.com/AaronLi-Alps/JqueryApiMigration.

The rest of this paper is organized as follows. Section 2 introduces related works, and Section 3 details the design of the API migration method. Section 4 presents the details of the implementation of the migration tool. Section 5 conducts experimental validation on 10 mainstream JS projects from

[1] VSCode is a code editor for building modern web. It can be accessed from https://code.visualstudio.com/.

DOI reference number: 10.18293/DMSVIVA24-136

GitHub and records the accuracy of the migration tool. Finally, Section 6 summarizes the paper and discusses future research directions.

II. Related Works

The use of third-party libraries is pervasive in modern software development [4]. Third-party dependencies are widely adopted across various programming environments and domains. However, the application of third-party dependencies introduces a range of dependency management issues, such as outdated versions and library conflicts. Research has found that most developers are reluctant to update the dependencies in their projects, and upgrading dependencies often leads to compatibility issues [5]. The primary reason is that the latest third-party libraries may not cover all the necessary features that the software references [6]. Issues related to third-party libraries are present in JavaScript, Python, and Java environments, each showing different degrees of these problems. Wu et al. [7] found that there are fewer occurrences of deprecated APIs in third-party libraries in Java ecosystem, whereas Python more frequently encounters API deprecation and replacement scenarios, and JavaScript exhibits the greatest degree of iteration changes, leading to more compatibility issues. Wang et al. [8] identified three main causes of disruptive API changes in open-source software libraries: changes in API signatures, changes in API implementations, and API deprecation and removal. These factors disrupt version compatibility during the library evolution process.

jQuery is one of the most widely used third-party libraries in the JS ecosystem. Due to frequent version iterations, most developers have adopted different versions of jQuery in different files. However, significant version updates in jQuery have led to the deprecation of many functions, presenting developers with the challenge of automatically modifying and handling deprecated functions caused by version upgrades. Existing tools for handling jQuery APIs do not fully automate jQuery version upgrades and require manual pre-processing to adapt to the migration methods of these tools. jQuery Migrate [9] is an official jQuery plugin designed to help users deal with disruptive changes by restoring the functionality of deprecated APIs and displaying warnings in the console. However, it is only a temporary solution for handling deprecated APIs during jQuery version updates, ultimately requiring developers to gradually address deprecated functions in their projects, failing to solve the issues arising from version updates completely. As a transitional solution for jQuery version migration, it still produces deprecated API incompatibility warnings, failing to completely resolve jQuery API version upgrade issues.

Dereck et al. [10] developed a migration plugin called unjQuery, which can replace jQuery API invocation with equivalent modern browser APIs but does not support jQuery API migration. Romulo et al. [11] conducted a study on deprecated APIs in JavaScript, categorizing deprecation types into four categories: utility deprecation, code comments, JS DOC comments, and console messages. Analyzing the top 320 most popular JS open-source projects on the npm website, they found that the primary deprecation types adopted in JS libraries are console messages, project documents, and code comments.

III. Methodology

A. Problem Formulation

We first illustrate the versioning issues of jQuery through a case study. zTree, a multifunctional tree plugin library dependent on jQuery, is widely used for its excellent performance, flexible configuration, and multifunctional combination. Version 3.5.48 of zTree uses a very old version 1.4.4 of jQuery, possibly because zTree was developed early. This brings about a problem: when users integrating zTree into their projects use a jQuery version higher than 1.4.4, e.g. 3.0, it causes issues with the asynchronous loading of node data in zTree, resulting in console errors such as "bind is not a function" (see Fig 1). According to the official jQuery documents, this is because the bind method is deprecated in jQuery version 3.0. Thus, during the jQuery version upgrade, zTree encounters dependency issues that need manual handling of deprecated API calls. Currently, programmers need to manually check each place where jQuery is used to see if deprecated functions are called. If it is possible to automatically detect deprecated functions in the project following a jQuery API version upgrade and handle these issues automatically based on official documents, it will alleviate the development burden and promote programming efficiency.

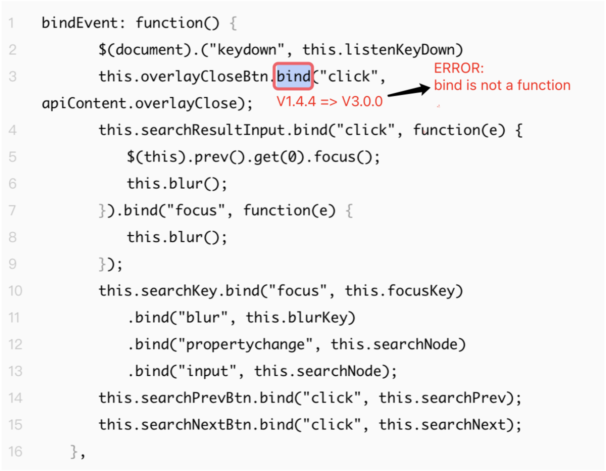
The problem of jQuery API automatic upgrade can be defined as follows: **How to automatically locate jQuery API usage within a project, determine if they are deprecated, and perform corresponding replacement and upgrade**?

B. Method Overview

Figure 2 describes the framework of the jQuery version migration tool. This migration tool takes project JavaScript files as input and comprises three modules: jQuery API document processing, API location and replacement, and API mapping algorithms. Addressing the questions posed in the previous section, this research investigates the automatic upgrade of jQuery APIs, designs an API mapping method for jQuery version upgrades, and implements a plugin for automatic jQuery version upgrades. The detailed steps are as follows:

(1) jQuery API Information Extraction and Processing. We collect records of modifications or deprecations for each API across different versions from the official jQuery documents, then form a usable dataset of API differences.

(2) jQuery API Location and Replacement. The Abstract Syntax Tree (AST) can decompose source code into a tree [12], effectively supporting code refactoring. By using a parser to parse JS files and generate ASTs, code location, refactoring, and replacement can be achieved. We also parse code to identify jQuery object references and their node positions, determine the used API methods through node parsing, and design an API replacement algorithm to replace jQuery APIs based on the mapping relationships obtained in Step (1).



```
bindEvent: function() {
      $(document).("keydown", this.listenKeyDown)
      this.overlayCloseBtn.bind("click",
apiContent.overlayClose);
      this.searchResultInput.bind("click", function(e) {
         $(this).prev().get(0).focus();
         this.blur();
      }).bind("focus", function(e) {
         this.blur();
      });
      this.searchKey.bind("focus", this.focusKey)
         .bind("blur", this.blurKey)
         .bind("propertychange", this.searchNode)
         .bind("input", this.searchNode);
      this.searchPrevBtn.bind("click", this.searchPrev);
      this.searchNextBtn.bind("click", this.searchNext);
   },
```

ERROR:
bind is not a function

V1.4.4 => V3.0.0

Fig. 1. Abandoned API caused by upgrading jQuery in Ztree.

(3) API Mapping. The mapping of deprecated APIs to new APIs is accomplished through a mapping set formed by analyzing API documents. After processing the API documents, a mapping set of jQuery APIs between different versions is generated. By iterating this API mapping set, one can check if the current API has a corresponding mapping relationship, thus obtaining the new API method that can be used as a replacement. .

C. jQuery API Information Extraction and Processing

In this section, we retrieve and process data on API changes from the official jQuery API website. API document is a crucial information source of third-party libraries, providing details on function changes and deprecations across different versions. We aim to collect API information from the jQuery official documents and use it as a basis for API migration.

API Information Analysis: By examining the official jQuery documents, it is possible to identify deprecated or modified API methods and determine the versions where changes occurred. These documents also include notes on deprecated functions, offering methods for handling them. For instance, the andSelf() API is marked as deprecated after version 1.8, with a note stating, "This API has been removed. Use .addBack() instead." This indicates that andSelf should be replaced with .addBack to avoid calls to deprecated APIs (see Fig. 3).

Data Retrieval: We utilize the BeautifulSoup library to retrieve API information, filtering out deprecated APIs from the official documents [13, 14]. The steps are as follows:

- Use BeautifulSoup to retrieve information from the general API page and individual API detail pages.
- On the general API page: obtain all function names and tags (including deprecation and abnormal status).
- On each API detail page: retrieve notes containing detailed information on API changes or descriptions.
- Save the data to a result file.

In total, we analyzed 318 jQuery APIs, identifying 40 as deprecated and 15 removed in updated versions. Among the 40 deprecated APIs, 11 provide corresponding solutions (new functions or alternative features). The API documents before version 2.0 often require users to perform feature detection, while 3.X or later versions provide replacement methods for user convenience.

D. jQuery API Location and Replacement

We analyze the API invocation of jQuery in JS files and use a JS parser to locate jQuery APIs and replace deprecated function APIs using an API migration algorithm.

Unlike Java language, JS lacks strict type definitions, which makes it difficult to locate jQuery methods precisely. We explore various methods to locate jQuery API invocations in projects by searching for jQuery references such as $. and $.bind. By converting JS files to AST structures with the Babel[2] tool, we process each file, modify the AST, and recompile it into JS code to replace the original file (see Fig. 4).

E. API Mapping

Algorithm 1: Obtain replaceable new APIs from jQuery documents.

```
Input : Two versions of API methods.
Output: API that changes between versions.
$info \leftarrow getNoteInf(oldMethodName)$;
if $info$ contains discarded or removed labels then
    if API document has a replaceable method then
        $newFunctionName \leftarrow getNewMethodName(oldMethodName)$;
        if Parameter changes then
            $getReplaceAPIWithParameter(oldMethodName)$;
            $newFunction \leftarrow getReplaceAPI(newFunctionName)$;
        end
        else $newFunction \leftarrow getReplaceAPI(methodName)$ ;
    end
    else
    No replaceable API found ;
end
```

We process jQuery API documents to obtain corresponding API mapping. API documents include suggestions for handling deprecated functions, commonly recommending replacements. We process such documents to derive new APIs for replacement.

An API replacement algorithm is proposed to utilize information from documents to obtain API mapping, shown in Algorithm 1. When a deprecated API is detected with a documented replacement, the algorithm maps old API parameters

[2] Babel is a JS compiler that supports processing JS ASTs. It can be accessed from https://babeljs.io/.

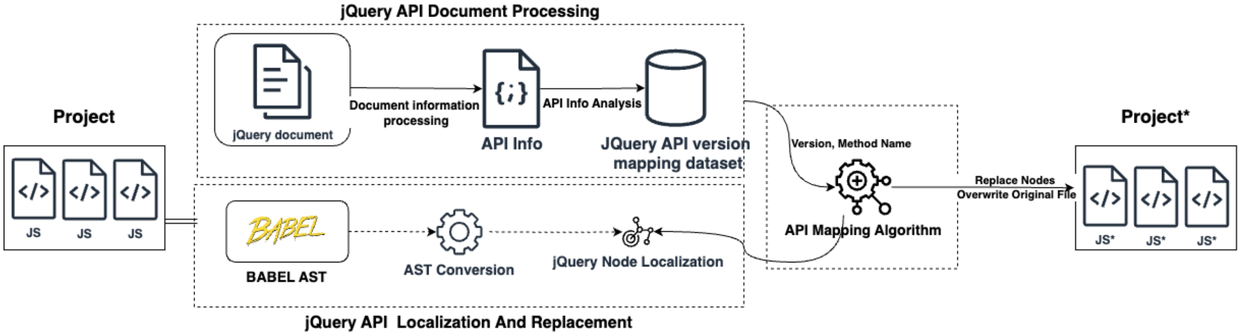
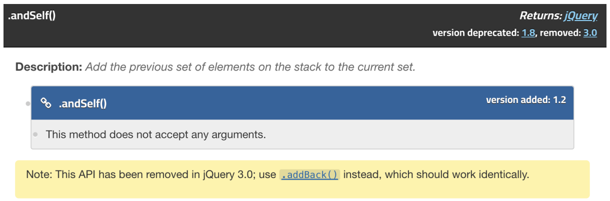


Fig. 2. The framework of our jQuery migration tool.


.andSelf()

Returns: jQuery

version deprecated: 1.8, removed: 3.0

Description: *Add the previous set of elements on the stack to the current set.*

.andSelf() version added: 1.2

This method does not accept any arguments.

Note: This API has been removed in jQuery 3.0; use `.addBack()` instead, which should work identically.

Fig. 3. Explanation of Deprecated API in jQuery API Documents.

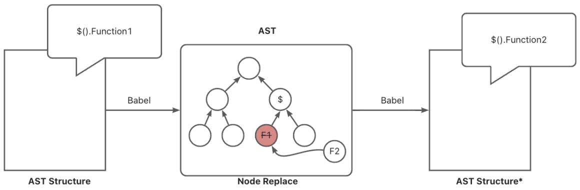


Fig. 4. Abstract Syntax Tree structure.

to the new one, replaces the API name, and rewrites the code in the original JS files (see Fig. 5).

To analyze jQuery API documents for replacement suggestions, we identify APIs marked as "removed" and their replacements. For example, the *andSelf()* function, first appears in version 1.8 and is deprecated in version 3.0. It should be replaced with *addBack()* as indicated in the documents.

The process includes parsing document notes to map deprecated functions to new APIs, storing the information, and forming a complete API mapping dataset to facilitate code replacement.

IV. Tool Implementation

This section integrates the above migration methods for the jQuery API and implements it as a Node.js-based command-line tool. It utilizes Babel[15] to upgrade jQuery from the existing version to the target version. The tool statically analyzes a given JS file, identifies deprecated jQuery APIs, and replaces them. It then generates a new AST structure, translates it back into JS code, and replaces the source file, effectively updating it to the new version of the jQuery API.

A. Tool implementation method

Figure 2 describes the main steps of the tool implementation. The specific implementation steps of the tool are as follows.

1. Construct AST from code. The migration tool retrieves the source code file and invokes the parsing module, which first parses the source code to construct an AST, and then examines the nodes in this AST tree.

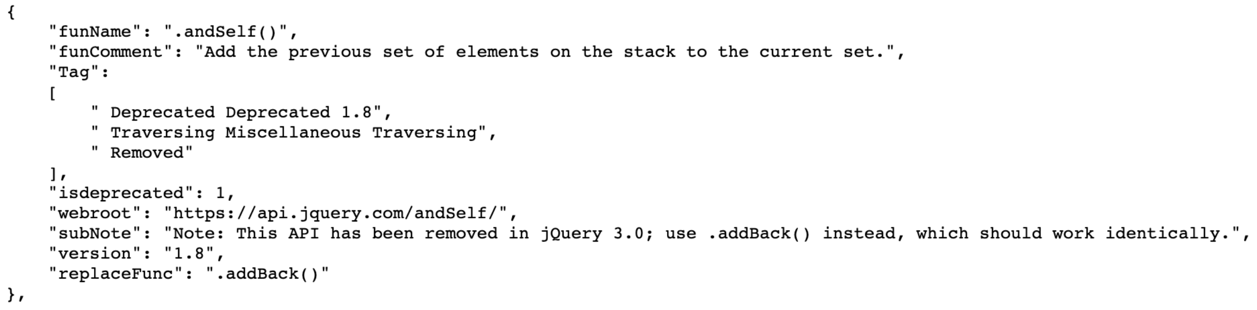
With Babel's code parser, the tool converts the code text into AST for analysis. The AST makes it easy to categorize the code by nodes, which is also applicable to weakly typed languages like JavaScript.

2. Locate the jQuery API invocations. The migration tool finds the jQuery API by searching the jQuery object as described in section 2.4. It utilizes Babel's visitor pattern, which is a method for AST traversal. This pattern involves an object that defines methods to identify specific nodes in a tree structure. By analyzing different types of nodes and their characteristics, the tool can extract specific information about each node in the code. To precisely locate the jQuery API in the source code, the visitor adds a search for distinctive symbols associated with jQuery nodes.

3. Replacement of the deprecated API invocations. After obtaining the node of the jQuery API, the function name within the node is judged and compared with the database obtained by processing the API documents in Section 2.3. If the API is marked as deprecated in the new version, it will be captured. Subsequently, the API mapping algorithm described in Section 2.5 is executed, and the function nodes with the deprecated API are replaced with the new version of the API node structure.

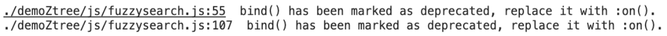
In the replacement process, the tool will replace the data (mapping pairs of the old and new APIs) also for structural parsing, the replacement content into function names, parameters, return values, and the corresponding set of node parser, according to the replacement type to replace the node of the AST structure.

4. Generate the target source code. After replacing the nodes with old APIs, the migration tool retrieves the complete AST structure of the source file and utilizes Babel to reverse parse the syntax tree into JS code, which is finally written into the source file.



```
{
    "funName": ".andSelf()",
    "funComment": "Add the previous set of elements on the stack to the current set.",
    "Tag":
    [
        " Deprecated Deprecated 1.8",
        " Traversing Miscellaneous Traversing",
        " Removed"
    ],
    "isdeprecated": 1,
    "webroot": "https://api.jquery.com/andSelf/",
    "subNote": "Note: This API has been removed in jQuery 3.0; use .addBack() instead, which should work identically.",
    "version": "1.8",
    "replaceFunc": ".addBack()"
},
```

Fig. 5. andSelf Replace Info.



```
./demoZtree/js/fuzzysearch.js:55  bind() has been marked as deprecated, replace it with :on().
./demoZtree/js/fuzzysearch.js:107  bind() has been marked as deprecated, replace it with :on().
```

Fig. 6. jQuery API upgrade tool result.

Ultimately, the tool identifies and replaces the API by inputting the source file code and the target jQuery version, abstract syntax tree parsing through Babel, and finally generates the replaced source code file. It parses the AST using Babel and then generates the modified source code file.

B. Tool application

This subsection explains how to use the tool. For projects that upgrade the jQuery API version, the tool can be run by configuring the script command. The script is run by declaring it in the scripts in the *package.json* file and setting the name of the script to run to *"jQapiUpgade": "node src/UpgradeJquery.js"*. The script can support adding the target version parameter to upgrade the project to the specified version of jQuery, add the *–version* parameter to the script, then the tool will accept the target version and replace the jQuery API with the specified version of the API during the API version upgrade process. After the script has been declared, the user can quickly execute the script with the help of the compiler or execute script commands in the console to upgrade the project's jQuery API.

After executing the command, the tool will automatically traverse the project's files to determine the jQuery API used in the project and automatically process the replacement API. if the tool finds that there is a deprecated API that needs to be replaced, it will output the location of the deprecated API and the information about the API that will be replaced on the command line and automatically replace the code. Figure 6 describes the console prompts for upgrading the zTree project with the API Upgrade Tool.

V. Experiments

In this section, we verify the capabilities of the jQuery API version migration tool through several experiments. We try to address the following research questions.

RQ1: Can the jQuery version migration tool accurately locate and replace jQuery APIs in actual projects automatically, and what is its accuracy?

RQ2: What are the differences between this jQuery version migration tool and other jQuery API version upgrade tools?

A. Experimental Datasets

To realistically and accurately evaluate the research, we select GitHub projects that depend on jQuery library, filtering to get those with a high star count (at least 1000 stars) and using older jQuery versions. This selection closely aligns with real project needs, as many projects stop updating due to jQuery version changes, which may cause some unexpected functionalities.

When selecting projects, we also check user feedback on the issues to see if users reported functionality issues due to jQuery version problems. This filter further verifies the necessity of upgrading the jQuery version. After these filtering criteria, we select 10 projects for experimental validation, performing actual jQuery API upgrades (see Table I). The "issue count" indicates the number of issues related to functionality problems due to jQuery version changes from the GitHub user issue list. For the tenth project DataTables, since the developer did not enable issue reporting, the issue count is unavailable, but a preliminary check detected deprecated API usage.

TABLE I
GitHub projects that include jQuery.

name	Star	jQuery Version	ISSUE
zTreev3	4k	1.4.4	5
bootstrap-table	11.6k	1.7.0	4
samizdatco/arbor	2.6k	1.6.1	5
jquery/jquery-ui	11.2k	1.8.0	3
OwlCarousel2	7.8k	1.8.3	8
Jquery-form/form	5.2k	1.7.2	3
jQueryAutocompletePlugi4	5.2k	1.4.4	8
tus-jquery-client	165k	1.9.1	4
Jquery-steps	1.7k	1.9.1	6
DataTables	7.2k	1.12.1	3

B. Experimental Process

We describe the experimental process of using the jQuery API migration tool on the zTree[3] project. The jQuery version in zTree is 1.4.4, and we aim to upgrade it to the latest version, 3.7. According to the method of handling jQuery API

[3] zTree is a multi-functional "tree plugins." based on jQuery. It can be accessed from https://github.com/zTree/.

documents described in Section 3.3, we identify the migration changes from version 1.4.4 to 3.7, finding 68 deprecated functions. Thus, the project might encounter up to 68 deprecated functions during the upgrade.

Next, we locate and replace jQuery APIs in zTree to see whether these 68 deprecated functions are used. Before replacing the jQuery APIs, if the console returns an error "xxx is not a function", it indicates deprecated function usage during the update.

Then, using the jQuery version migration tool, we search for all JS files using the jQuery API *bind()* method and locate all jQuery object references in all JS files by parsing them with AST. According to the steps for obtaining replacement API information described in Section 3.5, we find that 32 files in the zTree project contained references to the deprecated *bind()* function.

Finally, we validate the migration results. After the jQuery version migration, the zTree project runs normally, and previous error messages are eliminated. Functionality tests also confirm the normal operation, and we check the replaced parts to ensure proper functionality.

We then apply the jQuery API migration tool to the remaining 9 projects, upgrading their jQuery versions to the latest version, 3.7. Table II records the migration results for each project and lists the deprecated APIs detected by the migration tool. For 10 projects, our tool gets an average accuracy of 74.2% and solves 5 projects completely. Thus, we answer RQ1.

Note that for the third project *arbor*, some functions are similar to jQuery APIs, such as bind(). Although this function name is equal to the jQuery bind API, it is a customized method by programmers. The jQuery version migration tool can handle this by traversing the AST to determine whether the final call node object is a jQuery object, avoiding incorrect replacements. In the arbor project, certain upgraded functionalities required specific event triggers for validation. We wrote specific test cases to verify the functionality after the method migration. Proposing a general test method for verification would make the validation work more effective.

TABLE II
THE JQUERY VERSION MIGRATION AND UPGRADE RESULTS.

Project	Target Ver.	Current Ver.	Total	Find	Replace Acc.	Find Acc.
zTreev3	3.7	1.4.4	5	5	100%	100%
bootstrap-table	3.7	1.7.0	0	0	100%	100%
samizdatco/arbor	3.7	1.6.1	2	2	100%	100%
jquery/jquery-ui	3.7	1.8.0	4	4	75%	100%
OwlCarousel2	3.7	1.8.3	4	4	75%	100%
Jquery-form/form	3.7	1.7.2	8	7	100%	87.5%
jQueryAutocompletePlugin	3.7	1.4.4	10	8	87.5%	80%
tus-jquery-client	3.7	1.9.1	4	2	100%	50%
Jquery-steps	3.7	1.9.1	12	11	90.1%	91.6%
DataTables	3.7	1.12.1	3	5	100%	66.7%

C. *Experimental Results and Evaluation*

We compare the jQuery version migration tool with VSCode's built-in @type type declaration feature to verify the migration tool's accuracy and coverage. To verify the migration accuracy and compare the experimental methods, we conduct manual verification. Manual verification determines the usage of deprecated APIs in each project during the migration process and compares these findings with the tool's detection results. Manual verification is more accurate and can cover almost all areas involving APIs. Thus, we manually verified these 10 projects for API detection and handling during the jQuery version upgrade.

TABLE III
OUR TOOL VS. VSCODE: OBSOLETE FUNCTION DETECTION.

name	VSCode	Our Tool	Total APIs	Acc. (VSCode)	Acc. (Our Tool)
zTreev3	5	5	5	100%	100%
bootstrap-table	0	0	0	100%	100%
samizdatco/arbor	2	2	2	100%	100%
jquery/jquery-ui	4	4	4	100%	100%
OwlCarousel2	3	4	4	75%	100%
Jquery-form/form	6	7	8	75%	87.5%
jQueryAutocompletePlugin	16	8	10	50%	80%
tus-jquery-client	0	2	4	0%	50%
Jquery-steps	1	11	12	8.3%	91.6%
DataTables	2	5	3	66.7%	66.7%

The comparison results are shown in Table III. Our tool identifies and updates deprecated APIs used in the projects to a higher jQuery version, achieving an average accuracy of 74.2%, while VSCode gets an accuracy of 61.4%. It is remarked that the tool's accuracy will decrease in cases where it couldn't determine if the object is a jQuery object. The difference in replacement method accuracy mainly stems from developers rewriting or renaming API methods, causing the tool to incorrectly map nodes, leading to lower replacement accuracy.

VI. CONCLUSION

This paper proposes a tool for jQuery API version migration by processing API document information based on abstract syntax tree. It can automatically realize API processing after the jQuery version upgrade, fix the deprecated methods generated by the upgrade, and form API mapping based on the official documents. Through experiments on 10 popular projects from Github, the jQuery API version migration tool has an average accuracy of 74.2% in recognizing the deprecated jQuery API, which provides a better recognition range and accuracy compared with the @type plug-in for Visual Studio.

Since the research object in this paper is mainly jQuery, our tool has not been able to realize API migration for other third-party JS libraries. For future work, we hope to continue the research on version migration for other third-party libraries on JavaScript and realize version migration for multiple third-party libraries.

VII. THANK

This work is supported by NSFC(No.62272416) and the National Key Research Project of China(No.2023YFA1009402).

References

[1] W3Techs. *"Usage statistics and market share of JavaScript libraries for websites"*. https://w3techs.com/technologies/overview/javascriptlibrary/all.

[2] Danny Dig and Ralph Johnson. "How do APIs evolve? A story of refactoring". In: *Journal of software maintenance and evolution: Research and Practice* 18.2 (2006), pp. 83–107.

[3] Erik Derr et al. "Keep me updated: An empirical study of third-party library updatability on android". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 2187–2200.

[4] Parastoo Mohagheghi and Reidar Conradi. "Quality, productivity and economic benefits of software reuse: a review of industrial studies". In: *Empirical Software Engineering* 12 (2007), pp. 471–516.

[5] Raula Gaikovina Kula et al. "Do developers update their library dependencies? An empirical study on the impact of security advisories on library migration". In: *Empirical Software Engineering* 23 (2018), pp. 384–417.

[6] Li S et al. "Survey on Dependency Conflict Problem of Third-party Libraries". In: *Journal of Software* 34.10 (2023), pp. 4636–4660.

[7] Wei Wu et al. "An exploratory study of api changes and usages based on apache and eclipse ecosystems". In: *Empirical Software Engineering* 21 (2016), pp. 2366–2412.

[8] Wang Y et al. "Survey on Governance Technology of Open-source Software Library Ecosystem: Twenty Years of Progress". In: *Journal of Software* 35.2 (2023), pp. 629–674.

[9] Dave Methvin. *jquery-migrate*. https://github.com/jquery/jquery-migrate.

[10] Dereck J Bridie, Shinsuke Matsumoto, and Shinji Kusumoto. "unjQuerify: Migration of jQuery Snippets to Modern Vanilla JavaScript APIs". In: *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE. 2018, pp. 618–622.

[11] Romulo Nascimento, Eduardo Figueiredo, and Andre Hora. "JavaScript API Deprecation Landscape: A Survey and Mining Study". In: *IEEE Software* 3 (2022), p. 39.

[12] Jing Yu, Chenguang Mao, and Xiaojun Ye. "A novel tree-based neural network for android code smells detection". In: *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. IEEE. 2021, pp. 738–748.

[13] Resig John. *"jQueryAPI Document"*. https://api.jquery.com/.

[14] Resig John. *"jQueryAPI Detail Document"*. https://api.jquery.com/API_NAME/.

[15] Johannes Braams, Claudio Beccari, Javier Bezos, et al. *Babel*. Version 3.58. Available at https://babeljs.io/. 2021.

Automated Generation of Commit Messages in Software Repositories

Varun Kumar Palakodeti, Abbas Heydarnoori
Department of Computer Science
Bowling Green State University
Bowling Green, USA
{varunkp, aheydar}@bgsu.edu

Abstract

Commit messages are crucial for documenting software changes, aiding in program comprehension and maintenance. However, creating effective commit messages is often overlooked by developers due to time constraints and varying levels of documentation skills. Our research presents an automated approach to generate commit messages using Machine Learning (ML) and Natural Language Processing (NLP) by developing models that use techniques such as Logistic Regression with TF-IDF and Word2Vec, as well as more sophisticated methods like LSTM. We used the dataset of code changes and corresponding commit messages that was used by Liu et al. [12], which we used to train and evaluate ML/NLP models and was chosen because it is extensively used in previous research, also for comparability in our study. The objective was to explore which ML/NLP techniques generate the most effective, clear, and concise commit messages that accurately reflect the code changes. We split the dataset into training, validation, and testing sets and used these sets to evaluate the performance of each model using qualitative and quantitative evaluation methods. Our results reveal a spectrum of effectiveness among these models, with the highest BLEU score achieved being 16.82, showcasing the models' capability in automating a clear and concise commit message generation. Our paper offers insights into the comparative effectiveness of different machine learning models for automating commit message generation in software development, aiming to enhance the overall practice of code documentation. The source code is available at [2].

Index terms— Commit messages, Code documentation, Machine learning, Natural language processing, Automated commit message generation

DOI: 10.18293/DMSVIVA2024-145

1 Introduction

Commit messages play a vital role in software development, documenting changes to facilitate an organized workflow and effective collaboration. They offer insights into the what, why, and how of code modifications, aiding in tracking progress, understanding the development process, and supporting debugging and future enhancements [4, 7, 14]. The increasing complexity and scale of software projects underscore the necessity for commit messages, to ensure clear communication. Such documentation is crucial in open-source projects with diverse geographically contributors [7].

Crafting these messages manually in a fast-paced development environment and with varying documentation skills among developers leads to inconsistency, impacting code reviews, maintenance, and debugging [7, 17]. Moreover, under deadline pressures, developers might resort to generic descriptions, omitting crucial details [7]. These challenges highlight the potential of Machine Learning (ML) and Natural Language Processing (NLP) to automate commit message generation, aiming to enhance both consistency and quality of software documentation [5].

Significant efforts have been made in ML/NLP for automating commit message generation, including information retrieval and LSTM methods. NMT [9] uses a neural machine translation algorithm to translate code diffs into commit messages. NNGen [12] uses bag-of-words model and cosine similarity without a training phase. CoDiSum [19] and RACE [16] introduce tree-based neural networks and retrieval-augmented methods to improve message accuracy and relevance.

Despite their innovation, existing ML/NLP models like NMT, RACE, and CoDiSum face challenges such as high computational demands data limitations, and scalability issues, especially with Nearest Neighbors algorithms [9, 16, 19].

We chose conventional machine learning models over modern transformer-based approaches to ensure efficiency and lightweight operation. The primary reason was the ability of these ML models to run on the M1 chip and other small computational devices. This approach allows for effective commit message generation without the need for high-performance GPUs, making it more accessible and practical for various environments. By focusing on ML models, we aimed to balance performance with resource efficiency, ensuring that our solution remains both powerful and adaptable to a wider range of hardware configurations [8].

Our approach trains ML/NLP models on a dataset of code changes and corresponding commit messages to learn patterns crucial for generating effective commit messages. Using the same dataset used by Liu et al. [12], we divided the data into training, testing, and validation phases. Our experiments, especially the Cosine Similarity with TF-IDF and Nearest Neighbors algorithm achieving a BLEU score of 16.82, showcase our model's ability to align closely with actual commit messages.

This paper explores lightweight ML/NLP approaches for commit message generation, evaluating their efficiency, effectiveness, and architectural nuances. Through comprehensive comparative analysis and evaluations, including BLEU scores and manual assessments, we provide insights into the models' performance and the potential of automating commit message generation.

This paper is organized as follows: Section 2 details our approach, including data preparation and model training. Section 3 evaluates the proposed approach. Section 5 discusses the approach and results. Section 6 addresses threats to validity. Section 4 reviews related work. Section 7 concludes the paper and provides future research directions.

2 Proposed Solution

This section describes our approach, data preprocessing, data splitting, architecture, and training process of each model.

We focused on models like cosine similarity with TF-IDF, Word2Vec [3] combined with Logistic Regression, and the PyTorch LSTM Model due to their efficiency and adaptability in processing high-dimensional spaces and leveraging pre-trained embeddings for enhanced performance. The effectiveness of these selected models and the challenges encountered with others such as XGBoost, SVM, Multinomial Naïve Bayes, and DistillBERT, which either fell short in scalability or were resource-intensive, are discussed comprehensively in subsequent sections.

Table 1 shows the categories of the approaches and highlights the algorithms used for each approach.

Table 1: Algorithms and Categories of Approaches

Category	Algorithm/Approach
Similarity-Based	K-Nearest Neighbors, Cosine Similarity with TF-IDF
Ensemble methods	XGBoost
Support Vector Machines	SVM
Probabilistic models	Multinomial naïve bayes
Neural Network models	Simple RNN with MPS, PyTorch LSTM Model, DistillBERT with Hugging Face Transformer
Vector Space models	Word2Vec with Logistic Regression, Logistic Regression with TF-IDF
Combined Similarity and Vector Space	KNN with TF-IDF

2.1 Overview of the Proposed Solution

Our solution integrates machine learning and natural language processing to craft commit messages, focusing on improved data preprocessing and model architecture. We aim to surpass the limitations of models like NMT, NNGen, RACE, and CoDiSum by enhancing accuracy, efficiency, and handling complexity, using standard evaluation metrics for performance comparison.

2.2 Data Gathering

We utilized the dataset referenced in Liu et al. [12], originally compiled by Jiang et al. [9], a widely used benchmark in the field for evaluating new and existing methods.

2.3 Data Preprocessing

Our data preprocessing includes normalization steps such as whitespace stripping, lowercasing, tokenization, special character removal, stop word elimination, lemmatization/stemming, and vectorization. Except for LSTM, all machine learning models load data via a load_data(path) function, with preprocessing handled by built-in Python functions and the NLTK library.

For LSTM, we wrote a custom script for preprocessing to convert text to indices and build a vocabulary. This involved tokenizing, lowercasing, removing special characters and stop words, and tagging the start and end of sentences. We then divided the data into training and test/validation sets.

A separate vocab.py script generated a vocabulary file, vocab.txt, ranking tokens by frequency and incorporating special tokens. Another script standardized input lengths to

100 tokens by padding or trimming, using the ¡PAD¿ token as filler. The processed, padded files serve as the final input for LSTM model training and testing.

2.4 Model Architecture

Our models, leveraging Word2Vec embeddings and TF-IDF features, adopt straightforward architectures. The Cosine Similarity model, loads the data (training code changes and commit messages) and converts it into numerical vectors using TF-IDF.The model, trained on the Cosine Similarity principle, identifies similar vectors and generates commit messages by selecting the nearest vector from the training data [2].

In contrast, our Logistic Regression model with TF-IDF processes the data through a TF-IDF vectorizer and uses encoded commit messages for training. Implemented with PyTorch, it employs CrossEntropyLoss and Adam optimizer on Apple Silicon M1 GPU. This model outputs commit messages as text files. Both models are compared in Table 2.

The Logistic Regression model takes minutes to train and generate output, while the Cosine Similarity model takes seconds due to different learning techniques.

For Word2Vec, we use a distinct vectorization approach. Using Google's Word2Vec trained on Google News data, we employ a Cosine similarity-based Nearest Neighbors algorithm. The data undergoes vectorization to generate word embeddings, and the nearest vector match is used to generate commit messages. We also experimented with data preprocessing using the NLTK library for stop word removal, tokenization, and lower casing before vectorization.

To address scalability, we developed a Logistic Regression model integrated with Word2Vec, using similar preprocessing steps plus lemmatization. This model follows the same methodology as the TF-IDF model but uses Word2Vec embeddings.

Execution times vary, with Nearest Neighbors taking seconds and Logistic Regression taking minutes. Word2Vec offers extensive vocabulary and relies on a pre-trained model. A comparative analysis of Word2Vec-based models is given in Table 3.

Our LSTM-based model, built using PyTorch, is time-consuming and resource-intensive. Given the sequential nature of code changes, LSTM and Transformer models are ideal for capturing complexities in commit message generation. Preprocessing involves tokenization, stop word removal, and lemmatization. The data is then divided into training, testing, and validation sets, and standardized to a fixed length. The model's architecture is shown in Fig. 1

The LSTM model's Encoder processes the embeddings, creating a hidden state transferred to the Decoder, which generates commit messages sequentially. Training the LSTM model takes hours, with output generation taking minutes.

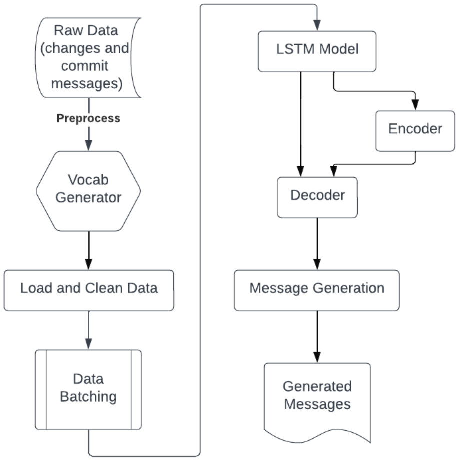


Figure 1: The architecture of the NLP model with the LSTM approach

3 Evaluations and Results

In this section we present an evaluation of the models, incorporating both qualitative and quantitative analyses to assess their effectiveness in generating commit messages.

3.1 Research Questions

Our study aims to answer the following research questions,

RQ1: Which ML/NLP method is best for creating commit messages? We will assess each model, comparing their BLEU scores and conducting manual evaluations.

RQ2: How well do ML/NLP methods reflect human understanding of code changes? Through manual reviews, we aim to gauge how these generated messages align with human perceptions and identify which outputs are preferred.

RQ3: How do Large Language Models (LLMs) like ChatGPT compare with traditional ML/NLP methods? We will prompt ChatGPT with code changes and compare its commit messages with those from our models in manual evaluations.

RQ4: Can simpler, quicker methods compete with advanced models that require more resources? We will test our streamlined models, designed for efficiency on the Apple M1 chip, against more complex, resource-intensive approaches.

Table 2: Comparison of TF-IDF based models

Feature	Logistic Regression with TF-IDF	Cosine Similarity with TF-IDF
Learning Type	Supervised Learning	Unsupervised Approach
Model Algorithm	Logistic Regression (PyTorch)	Cosine Similarity with Nearest Neighbors
Training Process	Trains on numerical labels of input training data	No training is required, training data for the similarity model
Hardware-utilized	Apple Silicon (mps)	Any CPU
Prediction Mechanism	Classifies instances into categories	Finds the nearest neighbor based on cosine similarity
Model Complexity	Involves weight adjustments and learning	Based on proximity in vector space
Interpretability	Can interpret feature importance	Straightforward (based on closest match in vector space)

Table 3: Comparison of Word2Vec based models

Feature	Logistic Regression with TF-IDF	Cosine Similarity with TF-IDF
Learning Type	Supervised Learning	Unsupervised Approach
Model Algorithm	Logistic Regression (PyTorch)	Cosine Similarity with Nearest Neighbors
Training Process	Trains on numerical labels of input training data	No training is required, training data for the similarity model
Hardware utilized	Apple Silicon (mps)	Any CPU
Prediction Mechanism	Classifies instances into categories	Finds the nearest neighbor based on cosine similarity
Model Complexity	Involves weight adjustments and learning	Based on proximity in vector space
Interpretability	Can interpret feature importance	Straightforward (based on closest match in vector space)

3.2 Evaluations Setup

Our experiments were conducted in a Python environment with all necessary libraries specified in the requirements.txt file of our repository [2]. We used a MacBook Air with an M1 chip for data preprocessing, model training, execution, and evaluation. PyTorch's MPS backend was used for leveraging the M1 chip.

3.3 Model Evaluation Criteria

We used the BLEU score to evaluate the performance of machine-generated text by comparing n-gram overlap between model output and human output [15]. We calculated the BLEU score using the script from Liu et al. [12], initially developed by Jiang et al. [9]. We also conducted manual evaluations, categorizing diffs into small (under 50 tokens), medium (50 to 75 tokens), and large (over 75 tokens) sets for detailed analysis.

In all of the papers and approaches that we have come across, BLEU, Meteor and Rouge-L are the most common metrics used for evaluation [16, 19]. However, Rouge-L metric is usually chose for approaches involving summarization tasks [11], our approach is deals with machine translation and requires a task that can prioritize precision and emphasize the establishment of the relevancy of the machine generated text to the original text on which the machine is trained on. Meteor, the other common metric is a computationally complex approach, that deals with flexible matching where synonyms and stemming is taken into account when judging the precision of text generated or comparing the machine generated text to original [10]. Hence, we use BLEU has higher accuracy when comparing the generated text to original as well as BLEU's approch penalizes when the words that are being compared are shorter than original [15].

3.4 Performance Results

Each model had a unique way of producing the output, but most output messages generated had overlaps with other models, the highest being 16.82. Table 4 can help under-

Table 4: BLEU scores of models

Model	BLEU Score
Cosine Similarity with TF-IDF and NN	16.82
Logistic Regression with TF-IDF	16.13
Cosine Similarity with W2V and NN	11.85
Cosine Similarity with W2V and NN (Preprocessed Data)	15.01
Logistic Regression with W2V and Preprocessed Data	3.17
LSTM Model	0.68

stand the comparative scores of the models.

Model evaluation was based on their ability to generate commit messages for the test set of code changes. Each model's unique approach led to variations in performance as shown in the table.

For manual evaluation, a script selected random diffs, and we manually ranked the generated commit messages. Comments were provided for each diff which explains the choice.

3.5 Quantitative Results

The Cosine Similarity model with TF-IDF and Nearest Neighbors achieved the highest BLEU score of 16.82 and the Logistic Regression model with TF-IDF also performed well with a BLEU score of 16.13.

In contrast, models using Cosine Similarity with Word2Vec varied based on preprocessing, without preprocessing they scored 11.85, and with preprocessing improved to 15.01. The Logistic Regression model with preprocessed Word2Vec data scored 3.17. The LSTM model achieved a BLEU score of only 0.68, reflecting its limitations.

3.6 Qualitative Results

Our qualitative evaluation delved into the interpretation of commit messages generated by various machine learning algorithms, including LSTM and LLM technologies. Our qualitative evaluation categorized diffs into small, medium, and large sizes to assess model performance.

Small Diffs (Under 50 Tokens): The Logistic Regression model with TF-IDF consistently delivered outputs that closely matched the actual commit messages, often outperforming the more detailed LLM outputs.

Medium Diffs (50 to 75 Tokens): LLMs like ChatGPT generated commit messages that captured semantic depth, often surpassing other models, but Logistic Regression with Word2Vec model generated outputs that were aligned with actual messages.

Large Diffs (Over 75 Tokens): LLMs provided demonstrated contextual understanding of the code changes, although sometimes exceeded the desired brevity. They accurately identified specific components within diffs. All the other methods did not produce good commit messages.

4 Related Work

In this section, we analyzed the related work in the field of code documentation practice, focusing on and committing message generation for source code changes. We have outlined our understanding and analysis of the research work that was explored.

4.1 Revisiting Learning-based Commit Message Generation

Dong et al. [6] underscored the importance of commit messages in software development, analyzing various methods for automated generation, particularly learning-based techniques. Their study examined the influence of datasets and model components on output quality, comparing rule-based, information retrieval-based, and learning-based approaches. They noted the advancements in deep learning have enhanced these techniques, though traditional models may obscure code semantics, leading to subpar performance. To address this, they introduced a two-stage generation paradigm that first creates an abstract representation of the commit message, with details added subsequently, thereby improving message relevance and clarity.

While Dong et al. [6] proposed a two-stage generation process, our research implemented and evaluated different models that utilize TF-IDF, Word2Vec, and LSTM in the context of commit message generation. This not only demonstrates the practical application of their theoretical insights but also provides a comparative analysis of various models, contributing to a broader understanding of ML/NLP applications in software documentation. Our research stands out by offering empirical evidence on the effectiveness of these models and adding to the existing knowledge base in automated commit message generation.

4.2 Model Architecture

Jiang et al. [9] utilized a Neural Machine Translation (NMT) approach, leveraging an encoder-decoder architecture, to convert diffs into commit messages. They compiled a substantial, quality-controlled commit dataset from sizeable projects and trained their NMT model on this data [9].

NNGen uses a bag-of-words model to convert diffs into vectors, then applies cosine similarity to find the most similar diffs using the Nearest Neighbors algorithm in the training set, streamlining the commit message generation process [12].

CoDiSum (Code Difference Summarization), a tree-based neural network model proposed by Shengbin Xu et al. [19], addresses limitations of prior commit message generation methods, particularly in understanding code structure and handling out-of-vocabulary (OOV) issues. CoDiSum's encoder captures the syntactic framework and semantic representation of code, identifying and replacing code identifiers with placeholders. This process helps in generating more accurate commit messages. The decoder, a multi-layer unidirectional GRU, uses attention weights and a context vector to generate commit messages word by word, integrating a copying mechanism for OOV words. This mechanism calculates a distribution over the structure sequence, allowing the model to directly copy specific terms from the code, thereby maintaining technical accuracy. CoDiSum combines generation and copying probabilities, using a sigmoid function to determine the final word choice in the commit message.

The RACE (Retrieval-Augmented Commit Message Generation) model is a two-module approach for generating commit messages from code changes. The model utilizes Information retrieval of similar commits and a Neural Network approach for the generation of commit messages. The first phase is information retrieval where the model identifies the most similar code change from a large dataset using Cosine similarity [16]. The dataset consists of around a million pairs of code changes and corresponding commit messages. Once the information retrieval phase successfully finds a similar diff, it is passed to the second phase, where the encoder utilizes a Transformer-based architecture, Feed Forward Network. Based on the similarity of the input and the retrieved diff - commit message pair, the model decides how much the example should influence the new messages and produces an encoding, that will be passed to the decoder. The decoder, which is also a Transformer-based component, generates the commit message token by token. To generate the next token the model takes into consideration of the previous token to keep the commit message tokens meaningful [16].

The models using the encode-decoder seq2seq approaches are complicated and require high-performance computational capacity compared to simple and lightweight approaches like ours. CoDiSum relies on understanding the semantics of the code changes, but our approaches are independent of the semantics so when our models encounter a new code change we base our output on the most similar vector whereas CoDiSum puts its efforts into understanding the code change's semantics if the semantics and logic of the code are complex the right commit message may not be generated. The major disadvantage of RACE is the reliance on information retrieval for commit message generation, our models in comparison are much more agile to any code change as they depend on vector similarity, and all of our models are straightforward and potentially faster to implement than RACE's two-module system.

4.3 Data Preprocessing

Data preprocessing is a critical phase in commit message generation, with each model adopting distinct but sometimes overlapping strategies. The NMT model focuses on cleaning commit messages by extracting relevant lines, removing identifiers, and avoiding large diffs, ensuring meaningful tokenization. CoDiSum goes further by extracting code structure and semantics, replacing identifiers with placeholders, and employing a copying mechanism for better message accuracy by including out-of-vocabulary words. NNGen simplifies diffs into vectorized "bags of words," prioritizing term frequency over syntax or order. RACE's unique preprocessing uses token-level actions to depict code changes, utilizing tags to emphasize code modifications.

Compared to these approaches ours involves more thorough data cleaning processes, ensuring the removal of irrelevant or noisy data which is critical for the quality of the input. sophisticated tokenization strategies that go beyond simple white-space or punctuation-based methods. In the case of word2Vec, it inherently understands the syntactic and semantic aspects of programming languages. Our preprocessing steps are optimized to complement the architecture of your chosen ML models, enhancing their learning efficiency.

5 Discussion

Our models demonstrated unique performance in generating commit messages, with notable overlaps. The TF-IDF Logistic Regression model generally produced outputs closer to actual commit messages compared to others, including the Word2Vec Cosine Similarity model, where data preprocessing showed limited impact.

To answer our RQ1, regarding the ML/NLP method that is best for creating commit messages, the TF-IDF Cosine Similarity model led in BLEU scores, indicating strong performance in generating commit messages. However, manual evaluations involving the qualitative analysis revealed the TF-IDF Logistic Regression model produced more accurate commit messages for certain diffs as for the LSTM model, given the available data and computational resources, may not be the most effective approach.

Concerning RQ2, regarding how well do ML/NLP models reflect human understanding of the code changes, the interpretation of code changes varies significantly among individuals, often not aligning with actual or generated messages. Yet, for medium and small diffs, some model's outputs did match human interpretation. For example, the model using the TF-IDF with Logistic Regression has been noted to match the understanding of code changes in the same way that the authors of the original commit messages.

Regarding RQ3, Large Language Models (LLMs) like ChatGPT, outperformed all ML and NLP models in manual evaluations, demonstrating a human-like understanding unmatched by traditional approaches.

For RQ4, dealing with the conundrum of whether simple and quicker methods compete with advanced models that require high computational power, we want to emphasize that even though RACE model holds the highest BLEU score (25.66), our TF-IDF Cosine Similarity model surpassed other approaches like NMT, NNGen, and CoDiSum in BLEU score performance, achieving 16.82, yet RACE remains the top performer in this field. Hence, simple approaches that can be run locally cannot be dismissed when it comes to generation of commit messages.

Table 5 shows the comparison of BLEU scores and compares the BLEU scores of the models that we designed to approaches that were contributed earlier to this domain. Table 5 does not include the pre-trained models. The BLEU score for ChatGPT could not be included as it was not released.

Also, over the course of our research through the last year, there have been significant efforts in leveraging LLMs like ChatGPT for commit message generation. Researchers have explored various aspects of using ChatGPT, and reviewing their work, we have made some observations such as the context-aware, superior performance of ChatGPT. In 78% of the evaluated samples, commit messages generated by ChatGPT were rated the best by human participants when compared to human-written commit messages, demonstrating their ability to produce high-quality, contextually accurate messages [20]. Although LLMs are highly capable, their accessibility and security concerns are reasons to prefer alternatives where the models are deployed locally, like our approach. Considering a scenario where an enterprise environment that spans across the globe requires implementing a model for enhancing their code documentation practice, LLMs like ChatGPT would need access to proprietary information to generate commit messages. In this scenario, there would be a higher preference for a model that runs locally.

Even though a model can run locally, for example the Llama developed my Meta and the variants of Llama LLM were primarily developed to run locally, but its entire State Dictionary that is publicly available consists of billions of tokens [18], making the model more generic to be an LLM, rather than our approaches that focus primarily on commit messages and code changes and are built on data gathered for this purpose.

Table 5: Comparison of BLEU scores of models

Model	**BLEU Score**
RACE [16]	25.66
Cosine Similarity with TF-IDF and NN	16.82
NNGen [12]	16.42
Logistic Regression with TF-IDF	16.13
Lucene [1]	15.61
NMT [9]	15.52
Cosine Similarity with Word2Vec and NN (Preprocessed Data)	15.01
Cosine Similarity with Word2Vec and NN	11.85
CommitGen [13]	14.07
CoDiSum [19]	13.97
Logistic Regression with Word2Vec and Preprocessed Data	3.17
LSTM Model	0.68

A significant advantage of LLMs that was observed in our research and the research efforts of Zhang et al. [21] was that the LLMs outperformed all the traditional models when the code diff size is large, i.e. when the tokens in the code diff exceeds 100 tokens, as pointed out in our evaluation of the large category of diffs. Zhang et al. [21] also notes that large diffs are not a majority of the cases when it comes to code changes. Also, there is a significant gap in real time adaptation of LLMs for code documentation as noted by the researchers.

6 Threats to Validity

In this section, we scrutinize the potential threats to the validity of our research, divided into internal validity, external validity, and reliability, to better understand their impact and strategize mitigations.

6.1 Internal Validity

One significant threat in our study is the integrity of the commit message data. If the developer's commit messages in the Java repos- itories we collected do not accurately reflect the associated diffs, our models could be trained on misleading data. The vast number of commit messages makes it unfeasible to manually validate each one for accuracy and relevance to its diff. Any discrepancies between the commit messages and the actual code changes could lead to models learning incorrect patterns, thus affecting the quality of the generated commit messages. But our source for the dataset is the same as the published work of Liu et al. [12] whose work, in turn, uses the dataset from Jiang et al. [9]

6.2 External Validity

Our research is bound to the Java programming language, chosen for its ubiquity and prevalence in software development [9]. This choice means our trained models are fine-tuned to the patterns and idioms of Java and may not generalize well to other programming languages without additional modifications. This language-specific focus presents a limitation in applying our findings to the broader field of automated commit message generation across diverse programming languages. To enhance the external validity, future work should consider incorporating datasets from various programming languages. This expansion would necessitate adjusting preprocessing routines to accommodate different syntactic and semantic structures, as well as retraining the models to recognize and process language-specific constructs accurately.

6.3 Reliability

Reliability concerns the reproducibility of our findings, manual evaluations can introduce subjectivity and bias. In our case, the evaluators are proven to show a significant level of experience and expertise with the Java programming language. Their review can be found in the repository [2].

7 Conclusions and Future Work

Generating commit messages poses a significant challenge, necessitating messages that are not only comprehensible to humans but also accurately reflect the context of code changes. Our investigation into various machine learning algorithms revealed that the Cosine Similarity with Nearest Neighbors and Logistic Regression algorithms are notably effective and computationally efficient for this purpose. However, their performance excels predominantly with smaller diffs, aligning closely with human interpretations of code changes. In contrast, Large Language Models (LLMs) like ChatGPT demonstrated superior performance for larger diffs. Our attempt to develop a lightweight LSTM model, despite being tuned for better performance, fell short of the effectiveness seen in models utilizing high-performance GPUs. Interestingly, the model with the highest BLEU score did not always surpass those with lower scores, especially within the realm of smaller diffs, where the Logistic Regression with TF-IDF vectorizer showed superior results.

Moving forward, we aim to broaden our exploration into commit message generation, considering the potential of various pre-trained models to enhance our approach. The objective is to develop a model that deeply understands the nuances of diffs in a human-like manner, effectively utilizing new and untrained words within the diffs. The ideal generated message should be concise, not exceeding 72 characters, yet meaningful. Additionally, optimizing the preprocessing of diffs for model training and focusing on accurately capturing the “what” aspect of changes remain priorities. This approach seeks to improve the model’s preference among human evaluators, ensuring it is adaptable for training on both CPU and GPU environments.

We shall explore modern technologies and LLMs like ChatGPT, Mistral AI and Meta-Llama and others. We have previously seen the potential of ChatGPT during out manual evaluation. We should explore models that have a potential for contextual awareness as to understand the semantic context of the code change that has been made, and generate an appropriate output in the form of commit message.

References

[1] Apache lucene. `https://lucene.apache.org/`. Accessed: 07-01-2024.

[2] Automated Generation of Commit Messages in Software Repositories. `https://doi.org/10.5281/zenodo.10888106`. Accessed: 07-01-2024.

[3] Word2Vec. `https://code.google.com/archive/p/word2vec/`. Accessed: 07-01-2024.

[4] R. P. Buse and W. R. Weimer. Automatically documenting program changes. In *Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering*, pages 33–42, 2010.

[5] J. Chen and Y. Xiao. Harnessing knowledge and reasoning for human-like natural language generation: A brief review. *arXiv preprint arXiv:2212.03747*, 2022.

[6] J. Dong, Y. Lou, D. Hao, and L. Tan. Revisiting learning-based commit message generation. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering*, pages 794–805, 2023.

[7] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen. BOA: A language and infrastructure for analyzing ultra-large-scale software repositories. In *Proceedings of the 35th IEEE/ACM International Conference on Software Engineering*, pages 422–431, 2013.

[8] M. Jafari, F. Majidi, and A. Heydarnoori. Prioritizing app reviews for developer responses on Google Play. In *Proceedings of the 30th International DMS Conference on Visualization and Visual Languages*, October 2024.

[9] S. Jiang, A. Armaly, and C. McMillan. Automatically generating commit messages from diffs using neural machine translation. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 135–146, 2017.

[10] A. Lavie and M. Denkowski. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23:105–115, September 2009.

[11] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the ACL Workshop: Text Summarization Braches Out*, January 2004.

[12] Z. Liu, X. Xia, A. E. Hassan, D. Lo, Z. Xing, and X. Wang. Neural-machine-translation-based commit message generation: How far are we? In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 373–384, 2018.

[13] P. Loyola, E. Marrese-Taylor, and Y. Matsuo. A neural architecture for generating natural language descriptions from source code changes. *arXiv preprint arXiv:1704.04856*, 2017.

[14] P. R. Mazrae, M. Izadi, and A. Heydarnoori. Automated recovery of issue-commit links leveraging both textual and non-textual data. In *Proceedings of the 37th IEEE International Conference on Software Maintenance and Evolution*, pages 263–273, 2021.

[15] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, 2002.

[16] E. Shi, Y. Wang, W. Tao, L. Du, H. Zhang, S. Han, D. Zhang, and H. Sun. RACE: Retrieval-augmented commit message generation. *arXiv preprint arXiv:2203.02700*, 2022.

[17] Y. Tian, Y. Zhang, K.-J. Stol, L. Jiang, and H. Liu. What makes a good commit message? In *Proceedings of the 44th IEEE/ACM International Conference on Software Engineering*, pages 2389–2401, 2022.

[18] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[19] S. Xu, Y. Yao, F. Xu, T. Gu, H. Tong, and J. Lu. Commit message generation for source code changes. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3975–3981, 2019.

[20] L. Zhang, J. Zhao, C. Wang, and P. Liang. Using large language models for commit message generation: A preliminary study. *arXiv preprint arXiv:2401.05926*, 2024.

[21] Y. Zhang, Z. Qiu, K. Stol, W. Zhu, J. Zhu, Y. Tian, and H. Liu. Automatic commit message generation: A critical review and directions for future work. *IEEE Transactions on Software Engineering*, 50(4):816–835, April 2024.

Prioritizing App Reviews for Developer Responses on Google Play

Mohsen Jafari*
Department of Computer Science
Bowling Green State University
Bowling Green, USA
mjafari@bgsu.edu

Forough Majidi*
Department of Comp. and Soft. Eng.
Polytechnique Montréal
Montréal, Canada
forough.majidi@polymtl.ca

Abbas Heydarnoori
Department of Computer Science
Bowling Green State University
Bowling Green, USA
aheydar@bgsu.edu

Abstract

The number of applications in Google Play has increased dramatically in recent years. On Google Play, users can write detailed reviews and rate apps, with these ratings significantly influencing app success and download numbers. Reviews often include notable information like feature requests, which are valuable for software maintenance. Users can update their reviews and ratings anytime. Studies indicate that apps with ratings below three stars are typically avoided by potential users. Since 2013, Google Play has allowed developers to respond to user reviews, helping resolve issues and potentially boosting overall ratings and download rates. However, responding to reviews is time-consuming, and only 13% to 18% of developers engage in this practice. To address this challenge, we propose a method to prioritize reviews based on response priority. We collected and preprocessed review data, extracted both textual and semantic features, and assessed their impact on the importance of responses. We labelled reviews as requiring a response or not and trained four different machine learning models to prioritize them. We evaluated the models' performance using metrics such as F1-Score, Accuracy, Precision, and Recall. Our findings indicate that the XG-Boost model is the most effective for prioritizing reviews needing a response.

Index terms— Prioritizing App Reviews, Mobile Applications, Machine Learning, Sentiment Analysis, App Stores.

1 Introduction

In recent years, the number of mobile apps in app stores, including Google Play, has increased dramatically. In the Google Play store, users can express their opinions and rate apps [1, 19]. These reviews include reports of problems and suggestions for app improvement. Also, app ratings indicate users' satisfaction and affect the number of downloads and the success of an app [16]. Studies show that the app rating is a principal factor for users. Apps with a rating of less than three stars are not downloaded by 77% of users [17]. To respond to user needs better, Google Play introduced a feedback mechanism for developers in 2013. To avoid receiving low ratings, developers must respond to user reviews. However, responding to reviews is time-consuming and costly, and less than 1% of reviews are responded to according [17]. Based on our knowledge, no research has been done on prioritizing user reviews to respond. Therefore, in this research, we have addressed the existing gap in prioritizing user reviews to respond. Two criteria are defined to measure the importance of reviews. The first criterion is whether or not to respond to reviews, and the second criterion is used to determine reviews that need to be responded to urgently (high-priority reviews).

The approaches presented in this paper include the following main steps. First of all, user reviews and developer responses are extracted from the Google Play store and are preprocessed. Secondly, textual and semantic features are extracted for each review and are measured the impact of each feature on the importance of the response to the review. Then we labeled our data based on response or not necessary response. After training four machine learning algorithms, their performances are evaluated. Finally, the

[*] These authors contributed equally to this work.

DOI: 10.18293/DMSVIVA2024-153

evaluations of several different apps with different properties indicate that the XGBoost algorithm shows the best performance.

The proposed approaches in this paper have the following main benefits for developers: (i) the volume of reviews that the developer has to read and respond to is reduced; (ii) Help developers decide which reviews to respond to; and (iii) developers are helped to determine the high priority reviews to respond.

The main contributions of this work include: (i) studying the overall relationship between textual and semantic features of reviews with developer responses, and identifying and introducing the most influential features in the importance of reviews. This aims to perform more accurate and comprehensive predictions of developer behaviour. Additionally, (ii) comparing the performance of four machine learning algorithms for solving the problem of prioritizing user reviews for responses, and selecting the best algorithm.

This paper is organized as follows. Section 2 presents the proposed approaches. Section 3 provides the evaluations of the proposed approach. Section 4 discusses the strengths and weaknesses of the proposed approach. Section 5 provides an overview of related work. Finally, section 6 concludes the paper and provides future research directions.

2 Proposed Solution

2.1 Approach Overview

We aim to develop a system for prioritizing user reviews to respond. The proposed solution involves several steps outlined in Figure 1. Initially, data is collected by selecting popular apps from Google Play and those referenced in previous research. Web crawling is utilized to extract initial information, which is then merged with existing datasets. In the preprocessing stage, reviews undergo various treatments such as language filtering, punctuation removal, and stemming. Textual and semantic features are then extracted in the third step to describe the reviews. Reviews are labeled based on whether a developer response is provided or not. Four machine learning algorithms are employed, including Decision Tree, Random Forest, Support Vector Machine (SVM), and XGBoost to model developer behavior. The impact of features on the output is assessed, and models are trained and evaluated for the best performance and result.

2.2 Data Collection

We aim to select datasets from a variety of app categories, such as photography, news, magazines, maps and navigation. In the first step, several popular apps on Google Play Store are identified. Then, all the information of these apps, including reviews and responses, is extracted by a web crawler. The collected data is merged with the dataset from the work presented in [7]. We gathered 431512 reviews that were responded to or not responded to. The dataset consists of the following 7 columns: App Name, Review, Review Score, Review Submission Date, Total Useful Score of the Review, Response, and Response Submission Date. The structure of the dataset is illustrated in Table 1

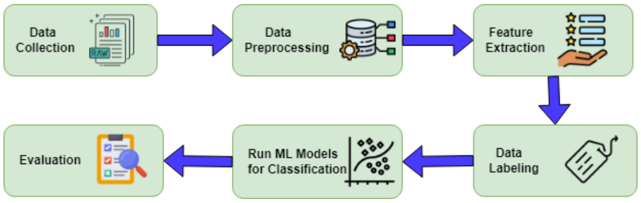


Figure 1: Approach overview

Table 1: The structure of the dataset

No.	Column Name	Description
1	App Name	The name of the unit that each app has
2	Review	User feedback about the app
3	Review Rating	The rating that the user gives to the app from 1 to 5
4	Review Submission Time	The date the user submitted their review
5	Total Helpful Votes	The number of people who found the review helpful
6	Response	The response that the developer registers for a review
7	Response Submission Time	The date the developer responded to the review

2.3 Data Preprocessing

The reviews are in natural language and are posted by individuals with different languages. Also, those who post these reviews may have spelling mistakes due to time constraints. Therefore, it is necessary to preprocess the input data. It should be noted that in this research, preprocessing has been done at four levels: character, word, sentence, and review. At the character level, punctuation marks are removed. Moving to the word level, stop words are eliminated, and letters become lowercase. Additionally, spell-checking is conducted, and roots are identified through morphological analysis, followed by general root finding. Single-letter words are then removed. At the sentence level, sentences are tokenized by breaking them down into individual tokens. Finally, at the review level, non-English reviews are removed, and reviews are segmented into sentences for further analysis. Most of the functions we use for preprocessing are from the NLTK library in Python.

2.4 Feature Selection and Extraction

In the field of machine learning, features are seen as measurable characteristics of an event. Selecting independent

Table 2: Textual and Semantic Features of Reviews

Feature No.	Feature Type	Feature Name
1	Textual	Clarity level of the review
2	Textual	Length of the review
3	Textual	Complexity level of the review
4	Textual	Number of nouns in the review
5	Textual	Number of verbs in the review
6	Textual	Number of adjectives in the review
7	Textual	Number of adverbs in the review
8	Semantic	Sentiment of the review
9	Semantic	Neutrality level of the review
10	Semantic	Polarity level of the review
11	Semantic	Rating of the review
12	Semantic	Total usefulness score of the review
13	Semantic	Informative level of the review
14 to 18	Semantic	Purpose of the review
19	Semantic	Title of the review
20 to 24	Semantic	Number of commitment-expressing words
25 to 34	Semantic	Review inclinations

and informative features is a fundamental step in using machine learning algorithms. After selecting suitable features to describe the event, they are prepared as feature vectors and fed into machine learning algorithms. It is necessary to transform the reviews into a format understandable by the computer. For this purpose, reviews need to be described using various features. These features are divided into two categories: textual and semantic which are introduced in Table 2.

2.4.1 Textual Features

Textual features provide us with information about the text of the review, but they do not provide any information about the content of the review. The *readability of review* will cause the developers to choose reviews that are more readable due to time constraints [12]. If the *review length* is long, the more likely it contains important information [10]. For example, the two review samples “bad” and “bad user interface, bad graphical design, the worst GUI.” are different in terms of their level of difficulty, which may affect how they are categorized by developers. Therefore, in this study, *review complexity* is chosen as a feature to describe the review. In [12], the total number of nouns and verbs is introduced as a feature to describe reviews on the social web. [3] also states that nouns and verbs convey the majority of the meaning of a text. Therefore, the *number of nouns in review* in each review is chosen as a textual feature to describe it. Also, there are considered textual features for the number of verbs, adjectives, and adverbs.

2.4.2 Semantic Features

Semantic features in this study are descriptors related to the content of reviews rather than the text itself. The features extracted include the sentiment of the review, neutrality, polarity, review rate, total helpfulness score, informativeness, review purpose, review title, number of words expressing commitment, and review tendencies. *Sentiment analysis* is crucial as users may give positive reviews with low ratings due to misinterpretation of how to rate an app correctly [7]. *Neutrality* and *polarity* levels are introduced as features characterizing reviews [12]. The *review rate* on Google Play, ranging from 1 to 5 stars, reflects user satisfaction and influences developer response likelihood [10]. The *total helpfulness score* indicates the perceived usefulness of a review. *Informativeness*, determined by TF-IDF, gauges the richness of opinions. *Review purpose* categorization includes informative, feature request, problem discovery, etc [26]. *Review titles* are considered significant descriptors [24]. The *count of commitment words* like “must” indicates persuasive elements [4]. *Review tendencies* encompass categories like anger, sadness, and positive/negative feelings, each quantified by word count [20]. The article [24] highlights that user reviews and developer responses have distinct sections with varying *Topics* and purposes, influencing the responses. On Google Play, users review diverse topics such as UI, security, and pricing. This research used the LDA (Latent Dirichlet Allocation) model to extract and categorize review topics. The LDA model, trained on different numbers of topics, found 10 to be optimal. The scikit-learn library in Python was utilized to implement the LDA algorithm and calculate topic assignment probabilities. These features collectively contribute to a comprehensive understanding of review content and aid in prioritizing them for response.

2.5 Data Labeling

This section explains the labelling process of the collected data. We use supervised learning methods, which require data to be labelled. Our main goal is to prioritize user reviews to respond. There are two approaches to do the labelling:

1. In the first approach, reviews that have been responded to by developers are given more importance and receive a label of 1, while reviews that have not been responded to receive a label of 0.
2. In the second approach, reviews that have been responded to more quickly (three days or less) by developers are given more importance and receive a label of 1, while reviews that have been responded to in a longer time frame receive a label of 0.

2.6 Identifying Influential Features

In this study, machine learning algorithms convert training data into feature vectors for model input [23]. The Pearson correlation coefficient is used to analyze the correlation between features and the output value, as well as the correlation among features themselves. If multiple features are highly correlated, redundant ones are ignored. The best model is then selected based on its performance in prioritizing reviews.

2.7 Machine Learning Models

In this study, four classification algorithms, namely Decision Tree, Random Forest, Support Vector Machine, and XGBoost have been used to learn developer's behavior and predict reviews that are of greater importance. Finally, a comparison has been made on the performance of these four models, and the best model has been selected. It is worth mentioning that the reason for selecting these four algorithms is their usage in other classification problems [2]. Additionally, these models represent the state of the art for this kind of dataset, with each employing a completely different algorithm. They are lightweight, resource-efficient, and have acceptable runtime execution. Once the classification algorithms were determined, we proceeded to train the models and evaluate their performance.

2.8 Prioritizing Reviews

This task is carried out from two perspectives. Firstly, it is examined whether the target review is worth responding to or not which we call Approach #1. For this purpose, after preprocessing and extracting textual and semantic features, machine learning methods are used to learn developers' past behaviour. The best model is selected for this perspective. Secondly, the importance of the review to the developer is indicated, and the decision is made on whether to respond quickly or not which is called Approach #2. After training the machine learning models, the best model for prioritizing reviews from this perspective is selected. This model assists the developer in deciding whether to respond to the review promptly or defer the response to another time.

2.8.1 Approach #1: Prioritizing Reviews as Requiring a Response or Not

In this approach, it is analyzed whether or not user reviews should be responded to. A dataset of 123,130 labeled reviews has been analyzed. After preprocessing and labeling the data, the textual and semantic features have been extracted from the reviews. Table 3 shows the correlation of each feature with the output variable in Approach #1.

After examining the correlation of each feature with the output variable and extracting the features that have the greatest impact on the output, we will train our models with selected features. Table 3 illustrates the features that we will use for Approach #1, highlighted in bold such as score, length, or the readability of the review.

Table 3: Abbreviated Name, Full Name and Correlation of Extracted Features with Output for Prioritizing Reviews for Both Approaches

Abbr	Feature Name	Approach #1	Approach #2
F1	Total Useful Review Score	0.031	-
F2	Review Score	**-0.51**	**0.34**
F3	Review Length	**0.22**	**0.12**
F4	Review Readability	**-0.16**	**0.2**
F5	Review Complexity	0.21	**0.13**
F6	Neutrality	**-0.12**	**0.56**
F7	Polarity of Review	**0.23**	**0.42**
F8	Number of Nouns	0.19	0.15
F9	Number of Verbs	0.026	0.33
F10	Number of Angry Words	0.025	0.36
F11	Number of Sad Words	0.007	0.36
F12	Number of Anxious Words	0.023	**0.36**
F13	Number of Negative Words	0.045	0.36
F14	Number of Positive Words	0.054	**0.39**
F15	No. of Words Commitment	-0.26	0.36
F16	Review Sentiment	0.022	0.27
F17	Topic 0	**0.12**	0.4
F18	Topic 1	0.099	0.41
F19	Topic 2	-0.021	0.41
F20	Topic 3	-0.023	0.4
F21	Topic 4	0.028	**0.4**
F22	Topic 5	**-0.17**	0.41
F23	Topic 6	0.087	0.42
F24	Topic 7	-0.015	0.4
F25	Topic 8	-0.063	0.41
F26	Topic 9	**0.12**	0.41
F27	Number of Adverbs	0.18	-
F28	Number of Adjectives	0.26	-
F29	Review Informativeness	0.033	-
F30	Feature Request Category	0.092	**0.36**
F31	Problem Detection Category	0.014	**0.36**
F32	Information Request Category	**0.13**	**0.36**
F33	Informer Category	**-0.17**	**0.38**
F34	Other Review Categories	**1**	**0.83**

Additionally, the features that are dependent on each other were also identified. It should be noted that, among the dependent features, the feature that has the highest correlation with the output variable is kept and the other features are removed from training the models. In this approach the features F3, F5, and F8 have strong correlations over 0.9, so we just consider F3 for our model.

2.8.2 Approach #2: Prioritizing Reviews as High or Low Priority for Response

In this approach, we determine whether or not a review needs to be responded to urgently (i.e., in less than three days) or not. For this purpose, data labelling is conducted differently from the previous section. Initially, a dataset of 308,382 reviews is earmarked for implementation and evaluation. Subsequently, reviews updated after receiving a response are removed due to a lack of initial post dates, leaving 284,062 reviews. The average response time to reviews of the dataset is calculated at 3.76 days, with reviews receiving responses within 0-3 days labelled as 1 and those after 4 days as 0. Data is then preprocessed before extracting textual and semantic features for each review. Notably, four features (useful votes, informative votes, adverbs, and

Table 4: Results of the performance of the four selected models for Approach #1

Model	Accuracy	F1 Score	Recall	Precision
Decision Tree	0.71	0.71	0.71	0.71
Random Forest	0.63	0.49	0.63	0.73
SVM	0.75	0.74	0.75	0.75
XGBoost	**0.77**	**0.78**	**0.77**	**0.77**

adjectives) are excluded. Pearson correlation coefficient is employed to calculate the correlation between features and output, as well as between features themselves. The correlation of each feature with the output variable is depicted in Table 3. Subsequently, our training models are evaluated using 5-fold cross-validation, ensuring that the data is split into five subsets, with each subset being used as a test set once while the remaining four subsets are used for training. No feature exhibits a correlation of less than 0.1 with the output variable. However, certain features show high interdependence, with features F8 to F11, F13, and F15 to F28 having correlations exceeding 0.9. Among them, only feature F21, with the highest correlation with the output variable, is retained, while the others are discarded.

3 Evaluations

This section represents the results obtained from training four machine learning models for two approaches which can be seen in Tables 4 and 5. The performance metrics we used in this research are Accuracy, F1-Score, Recall, and Precision.

3.1 Evaluations of Approach #1

In this section, the performance of the models trained in Section 2.8.1 will be evaluated.

After determining the features that had a correlation of more than 0.1 with the output variable and were also not dependent on any other feature such as review score, review length, and review readability, the selected models were trained. The first part of Table 4 shows the results obtained from the performance of each of the models when considering only the selected features.

Table 4 shows that the XGBoost model has a higher F1-Score than other models for this approach with a value of 0.77. The other performance metrics such as Accuracy with a value of 0.78, Recall with a value of 0.78, and Precision with a value of 0.77 have higher values as well. According to the results obtained in this section, it can be concluded that the proposed approach in this paper for prioritizing user reviews to respond from the aspect of whether or not to respond to a particular review works well and has achieved very good and acceptable results.

Table 5: Results of the performance of the four selected models for Approach #2

Model	Accuracy	F1 Score	Recall	Precision
Decision Tree	0.86	0.85	0.86	0.85
Random Forest	0.26	0.32	0.26	0.86
SVM	0.91	0.87	0.91	0.83
XGBoost	**0.91**	**0.87**	**0.91**	**0.86**

3.1.1 Case Study Evaluations

In this section, the best model obtained was tested on 10 popular applications. Firstly, ten popular applications from ten different categories were selected, and then 2000 comments were randomly selected for each application. After selecting the applications, the best model obtained was evaluated on these ten applications. The mean of F1-Score for theses 10 applications is 0.77 that shows the acceptable performance of the selected model. Also, it can be concluded that the presented model can be used for different types of applications from different categories.

3.2 Evaluations of Approach #2

In this section, the performance of the models trained in Section 2.8.2, which aim to prioritize reviews from the perspective of the importance of responding to them more quickly (3 days or less than 3 days), will be evaluated.

After determining the features that were not dependent on any other feature like review readability, neutrality, and polarity of review, the selected models were trained. The second part of Table 5 shows the results of the performance of each model.

Table 5 reveals that the proposed approach has the best performance with an F1-Score of 0.87 using the XGBoost algorithm. The other performance metrics of XGBoost have higher values than other models. The Accuracy of 0.91, Recall of 0.91, and Precision of 0.85 reveal the best performance of XGBoost among other models used in this research. It has also been observed that the proposed approach works well and can be used to help developers prioritize reviews for response.

3.2.1 Case Study Evaluations

In this section, the best-trained model is evaluated on 8 popular applications. Firstly, 8 popular applications from different categories are selected and the selected model is tested on 1000 to 1500 reviews from each of the applications. After running the best model, the mean of F1 Score for all applications was 0.87 which is acceptable performance of the selected model. Also, according to the result, it can be concluded that the presented model is applicable for different types of applications from different categories.

3.3 Threats to Validity

In this section, the threats to the validity of our results are discussed. These threats are categorized into *internal*, *external*, *construct*, and *reliability* threats.

- **Internal Validity**: The accuracy and performance of the proposed approach are influenced by the features considered (such as the number of sentences, and the devices on which reviews are recorded) and the tools and libraries used. To mitigate these threats, the study uses well-known and accurate tools selected carefully.
- **External Validity**: The threat here is the potential inadequacy of collected data. This is addressed by using a crawler to extract reviews from many programs and merging this data with existing data from another source [7].
- **Construct Validity**: There is a concern about whether the textual and semantic features considered are relevant to the output. This is tackled by calculating the correlation between features and the output variable using Pearson's correlation coefficient, demonstrating that most selected features positively impact the prioritization of reviews.
- **Reliability**: This examines the consistency of results with similar inputs. The study ensures reproducibility through steps like preprocessing, feature extraction, and model training. Despite the probabilistic nature of machine learning, the study selects a diverse range of reviews to achieve a comprehensive understanding and mitigate uncertainty. The source code and dataset of the research are available at `https://github.com/ISE-Research/App-Reviews-Prioritization`.

4 Discussion

The evaluation results show that the XGBoost model has the best performance for prioritizing reviews with an F1-Score of 0.77 and 0.87 from Approach #1 and Approach #2. Therefore, developers can use the models presented in this study to prioritize user reviews and spend less time and effort responding to them.

In the analysis of our dataset, which consists of multiple attributes including 'rate', 'review length', 'readability score', 'review subjectivity', and several others, the XGBoost algorithm demonstrated superior performance compared to SVM, Decision Tree, and Random Forest algorithms. This can be attributed to XGBoost's ability to handle complex data interactions and its efficient implementation of gradient boosting, which allows it to capture intricate patterns and relationships within the data. Unlike SVM, which can struggle with large datasets and non-linear separations, XGBoost effectively scales with the dataset size and complexity. Similarly, while Decision Trees can easily overfit and Random Forests, though more robust, may not achieve the same level of accuracy and fine-tuning, XGBoost's advanced regularization techniques and hyperparameter optimization make it more adept at producing high-precision, high-recall, and high F1-Score metrics. This results in a more reliable and accurate predictive model for our multi-attribute dataset, highlighting XGBoost's efficacy and robustness in handling diverse and intricate data structures.

Nevertheless, like any other method, this work also exhibits both strengths and weaknesses. In terms of strengths, it introduces a novel categorization of user reviews based on two distinct approaches, incorporating a comprehensive array of textual and semantic features. It also identifies a substantial number of features crucial for prioritizing reviews for response, offering a fresh criterion for evaluating developers' perspectives. The study's validity is reinforced by its analysis of reviews spanning various application categories, and it carefully evaluates four machine learning algorithms to determine the most effective one for review prioritization. However, there are also weaknesses to consider. For instance, we exclude reviews updated after receiving a response due to limitations in accessing the initial posted date on Google Play. If we have a larger dataset, we can use deep learning techniques to achieve better results. Additionally, we can enhance the dataset with more features to obtain more precise outcomes.

5 Related Work

The studies conducted on the reviews are classified into several important areas: examining the relationship between users and developers, responding to user reviews, user review features and developer responses, automated response generation, user review classification, user review feature extraction, user sentiment analysis, user review analysis, and other divisions. Our main focus is on studies that examine user reviews and developer responses.

Importance of Responding to User Reviews: The study [17] concludes that as the number of downloads of an application increases, the number of responses developers provide to user reviews decreases, and conversely, as the number of downloads decreases, the number of responses to user reviews increases. It has also been noticed that users who initially rated an app with one star upgraded their rating to five stars after receiving a response.

Automatic Response Generation: The works in [25], [7], [9] examine various techniques used for responding to user reviews in apps, including linguistic patterns and keyword similarities. These studies are based on two main approaches. In the first approach, a tool is provided to developers to help them respond to user reviews on Google Play. This tool is the first automatic tool that does not rely on pre-prepared or rule-based responses; instead, it uses neural networks to generate responses. The second approach focuses on automatic response generation using neural machine translation networks. The article [11] presents ChatReview, a ChatGPT-enabled NLP framework designed to analyze domain-specific user reviews, offering personalized search results and addressing challenges such as bias and privacy.

Feature Extraction from User Reviews: [13], [8], and [27] focus on the automatic extraction of features from user reviews on internet sites. Finally, [13] deals with the extraction and matching of features mentioned in app descriptions and user reviews. The approach presented in [8] investigates the factors that determine the perceived helpfulness of user reviews. Specifically, it explores the correlation between user innovativeness and the content of reviews for innovative products, utilizing data gathered from a questionnaire survey. [27] aims to extract feature-describing phrases from app descriptions, align each app feature with its corresponding user reviews, and construct a regression model to determine which features exhibit significant associations with app ratings. Finally, [1] proposes an approach that assists in augmenting labeled datasets of app reviews by utilizing information extracted from GitHub issues that contain valuable information about user requirements.

Sentiment Analysis of User Reviews: In [15], a solution for employing sentiment analysis tools in software engineering datasets is presented. This work examines sentiment analysis in Stack Overflow discussions, JIRA issue reviews, and Google Play reviews. [22] employs a pre-trained RoBERTa language model for sentiment analysis and calculates cosine similarity for content-based recommendations. In [5], the impact of deep learning on sentiment analysis of Chinese reviews is examined.

Users' Review Analysis: This section presents the work related to the analysis of user reviews on internet sites. The work in [21] deals with automatic review analysis using clustering, TF-IDF, vector space model. [6] focuses on automatic insight extraction from user reviews over some time and prioritizing them using Gaussian distribution, linear regression, correlation coefficient, moving average, and Pearson correlation. [26] presents a tool for user review analysis using Word2Vec, vector space model, TF-IDF, and moving average. [14] presents MApp-IDEA tool to identify and categorize emerging concerns from user reviews, organizing them into a risk matrix with prioritization levels, and tracking their evolution over time. Finally, [18] presents UX-MAPPER, an approach to analyzing app store reviews and helping practitioners in identifying key elements influencing user experience.

Previous studies have used various machine learning methods to classify features from user reviews effectively. However, they have not explored using machine learning to prioritize reviews for responses. This research addresses this gap by collecting extensive datasets, identifying effective textual and semantic features for responding to reviews, and utilizing machine learning methods for prioritization.

6 Conclusions and Future Work

In recent years, the number of mobile applications in mobile app stores has been increasing, and every day a large number of users post their reviews and rate applications. Therefore, these users expect to receive a response from developers. Sometimes users increase their ratings after receiving a response from developers. As a result, to increase the app rating, it is necessary to respond to user reviews. However, responding to the large volume of reviews is one of the challenges facing developers, and they are not able to respond to all the reviews they receive. Therefore, it is necessary to prioritize user reviews to determine whether a developer should respond to a review or not. If so, it is important to know the response urgency associated with each review. In this paper, we proposed an approach based on natural language processing and machine learning techniques to prioritize users' reviews to respond. According to our evaluations, we observed that the XGBoost yields the most favourable outcomes with the best F1-Score with the value of 0.77 for Approach #1 and 0.87 for Approach #2 among other trained models for prioritizing user reviews regarding whether a response is necessary. Additionally, the XGBoost model demonstrates superior performance in determining the priority level of user reviews for response.

In the future, we plan to delve deeper into analyzing more textual and semantic features, like the number of sentences in a review, mentions of specific topics, and the types of devices used for leaving reviews. It would also be beneficial to test out different machine-learning algorithms to see how they compare in terms of performance. Additionally, it is important to see how well our approach works specifically within the Apple Store. Another useful step would be to categorize and group together reviews that share similar content, making it easier for developers to respond more effectively. Lastly, with the increased popularity of Large Language Models (LLMs) and Generative AI, we plan to work on generating automated responses to reviews to streamline the process even further.

References

[1] Y. Abedini and A. Heydarnoori. Can GitHub issues help in app review classifications? *ACM Transactions on Software*

Engineering and Methodology, July 2024.

[2] N. Ailon and M. Mohri. An efficient reduction of ranking to classification. *arXiv preprint arXiv:0710.2889*, 2007.

[3] G. Capobianco, A. D. Lucia, R. Oliveto, A. Panichella, and S. Panichella. Improving IR-based traceability recovery via noun-based indexing of software artifacts. *Journal of Software: Evolution and Process*, 25(7):743–762, 2013.

[4] S. E. Crawford and E. Ostrom. A grammar of institutions. *American political science review*, 89(3):582–600, 1995.

[5] M.-Y. Day and Y.-D. Lin. Deep learning for sentiment analysis on Google Play consumer review. In *Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI)*, pages 382–388, 2017.

[6] C. Gao, J. Zeng, D. Lo, C.-Y. Lin, M. R. Lyu, and I. King. Infar: Insight extraction from app reviews. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 904–907, 2018.

[7] C. Gao, J. Zeng, X. Xia, D. Lo, M. R. Lyu, and I. King. Automating app review response generation. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 163–175, 2019.

[8] Y. Goto and R. Tsuchihashi. What content of user reviews is considered helpful? In *Proceedings of the 12th International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 490–494. IEEE, 2022.

[9] G. Greenheld, B. T. R. Savarimuthu, and S. A. Licorish. Automating developers' responses to app reviews. In *Proceedings of the 25th Australasian Software Engineering Conference (ASWEC)*, pages 66–70, 2018.

[10] S. Hassan, C. Tantithamthavorn, C.-P. Bezemer, and A. E. Hassan. Studying the dialogue between users and developers of free apps in the Google Play store. *Empirical Software Engineering*, 23:1275–1312, 2018.

[11] B. Ho, K. L. Nguyen, M. Dhulipala, V. K. Pallipuram, et al. ChatReview: A ChatGPT-enabled natural language processing framework to study domain-specific user reviews. *Machine Learning with Applications*, 15:100522, 2024.

[12] C.-F. Hsu, E. Khabiri, and J. Caverlee. Ranking comments on the social web. In *Proceedings of the International Conference on Computational Science and Engineering*, volume 4, pages 90–97. IEEE, 2009.

[13] T. Johann, C. Stanik, A. M. Alizadeh B., and W. Maalej. SAFE: A simple approach for feature extraction from app descriptions and app reviews. In *Proceedings of the 25th IEEE International Requirements Engineering Conference (RE)*, pages 21–30, 2017.

[14] V. M. A. d. Lima, J. R. Barbosa, and R. M. Marcacini. MApp-IDEA: Monitoring app for issue detection and prioritization. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*, pages 180–185, 2023.

[15] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto. Sentiment analysis for software engineering: How far can we go? In *Proceedings of the 40th International Conference on Software Engineering*, pages 94–104, 2018.

[16] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering*, 21:1067–1106, 2016.

[17] S. McIlroy, W. Shang, N. Ali, and A. E. Hassan. Is it worth responding to reviews? Studying the top free apps in Google Play. *IEEE Software*, 34(3):64–71, 2017.

[18] W. T. Nakamura, E. C. C. de Oliveira, E. HT de Oliveira, and T. Conte. Ux-mapper: A user experience method to analyze app store reviews. In *Proceedings of the XXII Brazilian Symposium on Human Factors in Computing Systems*, pages 1–11, 2023.

[19] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *Proceedings of the 21st IEEE International Requirements Engineering Conference (RE)*, pages 125–134, 2013.

[20] J. W. Pennebaker, M. E. Francis, and R. J. Booth. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001, 2001.

[21] M. V. Phong, T. T. Nguyen, H. V. Pham, and T. T. Nguyen. Mining user opinions in mobile app reviews: A keyword-based approach. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 749–759, 2015.

[22] L. L. Scientific. Sentiment-based recommendation for online shopping. *Journal of Theoretical and Applied Information Technology*, 102(9), 2024.

[23] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto. Comments on researcher bias: the use of machine learning in software defect prediction. *IEEE Transactions on Software Engineering*, 42(11):1092–1094, 2016.

[24] P. M. Vu, T. T. Nguyen, and T. T. Nguyen. On building an automated responding system for app reviews: What are the characteristics of reviews and their responses? *arXiv preprint arXiv:1908.10816*, 2019.

[25] P. M. Vu, T. T. Nguyen, and T. T. Nguyen. Why do app reviews get responded: A preliminary study of the relationship between reviews and responses in mobile apps. In *Proceedings of the ACM Southeast Conference*, pages 237–240, 2019.

[26] P. M. Vu, H. V. Pham, T. T. Nguyen, and T. T. Nguyen. Tool support for analyzing mobile app reviews. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 789–794, 2015.

[27] H. Wu, W. Deng, X. Niu, and C. Nie. Identifying key features from app user reviews. In *Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 922–932. IEEE, 2021.

Improved Gaussian Mixture Model for Feature Classification Based on Hypergraph Structure

1st Zhiwei Zhao
Dalian University of Technology
School of Computer Science and Technology
Dalian, China
1329065761@qq.com

2nd Qiang Zhang
Dalian University of Technology
School of Computer Science and Technology
Dalian, China
zhangq@dlut.edu.cn

3rd Kai Lin
Dalian University of Technology
School of Computer Science and Technology
Dalian, China
link@dlut.edu.cn

***Abstract*—In recent years, interest in deep learning methods based on nonlinear structures for natural language processing has significantly increased. At the same time, hypergraph convolutional neural networks have started to replace graph convolutional neural networks, allowing the integration of implicit information with the complexity of the processed data. However, these hypergraph-based methods face challenges, such as the inability to capture hidden relationships and the inaccurate calculation of feature similarity between structures. A recent study proposes a hypergraph structure based on probability distributions. This approach constructs hypergraphs around multi-label correspondences and introduces a convolutional update to the framework. The study's approach includes two main components: the hypergraph construction model based on the refined EM algorithm and the adaptive dynamic convolution model. The refined EM algorithm calculates feature similarity distance to dynamically create the hypergraph structure, enhancing structural integrity at each layer. Furthermore, the study introduces a dynamic convolution module consisting of two phases: vertex convolution and hyperedge convolution. This module combines the features between the vertices and hyperedges. The study evaluates the proposed method using the Cora and Meld datasets, demonstrating its superior performance compared to contemporary state-of-the-art methods. Additionally, supplementary experiments demonstrate the method's efficacy and resilience across diverse data distributions.**

***Index Terms*—hypergraph, hybrid Gaussian model, dynamic convolution**

I. Introduction

Graph structures, as extensions of linear relational representations, provide a powerful method to interpret and represent relationships between data using graph-theoretic methods [1]. Graph structures have had a significant impact on reinforcement learning and play a crucial role in the efficiency of representation learning. Traditional graph structures have limitations in capturing feature relationships, especially in multi-dimensional features, leading to suboptimal experimental results [2]. To address these limitations, a more advanced structure is needed to support representation learning in neural network models [3]. Researchers are turning to graph theory to extend simple graph structures to hypergraph structures, which can represent complex data relationships more effectively. As shown in Figure 1, unlike traditional graph structures, hypergraphs can connect multiple vertices, allowing for better representation of higher-order relationships in the data and more accurate descriptions of complex data structures and interconnections. Although the research field of hypergraphs is relatively new, their high adaptability in complex multi-dimensional data environments has already proven valuable in various applications. Combining hypergraph and neural network models can enhance the ability to handle relationships between data [3]. To address this issue, Jiang et al. proposed the dynamic hypergraph construction model DHGCN [4]. This model is divided into two parts, constructing and updating the network to improve the accuracy of the hypergraph structure. It outperforms existing methods on the Cora citation network and microblogging dataset and is robust across different data distributions. However, DHGCN focuses on improving model accuracy through dynamic convolution under the hypergraph structure, which ignores implicit attributes n the feature space. Additionally, using a uniform similarity projection space for feature mapping in the hypergraph structure loses relationships

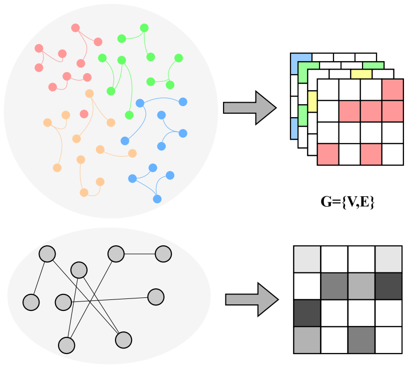


Fig. 1. The comparison between graph and hypergraph.

DOI reference number: 10.18293/SEKE2024-157

between objects and weakly correlated labels.

To address the above shortcomings, this paper proposes an improved hybrid Gaussian model to compute the similarity distribution. This model works with a hypergraph structure to dynamically convolve and update the multidimensional probability distribution [3]. In the similarity distribution calculation module, the hypergraph structure is constructed by iteratively obtaining the multidimensional probability distribution mapping using an improved expectation maximization algorithm. The dynamic convolution module computes adaptive weight transformation matrices to perform node convolution and hyperedge convolution, respectively. Compared to the DHGNN method, our approach embeds implicit relations into object features more effectively [4]. Additionally, adaptively changing the hyperedge weights ensures better integration of this embedding information into DHGN and global feature information. In this study, we apply our model to data with implicit relationships and perform label classification on the Cora citation dataset and the MELD multi-modal dataset.

The contributions of this paper are as follows:

1) We propose a hypergraph construction method based on an improved Gaussian mixture model to compute the similarity distribution. This method employs a sampling-optimized EM algorithm to compute the distribution of mappings from object features to the label space, embedding implicit relationships into the features.
2) We propose a probability distribution-based adaptive updating method, which uses probability distributions to compute the weights of hyperedges and update the hypergraph structure via vertex convolution and hyperedge convolution modules. This ensures that the hypergraph structure can fuse the implied relationships into the global features.
3) We conducted experiments on label classification using textual citation data. Our method demonstrates superior performance compared to current state-of-the-art approaches and exhibits enhanced resilience across various data distributions.

II. Related Work

A. Graph-based Deep Learning

Graph Neural Networks (GNNs) are neural network models designed for graph data. Their main purpose is to gain a deeper understanding of the nodes and edges in the graph structure to analyze and process graph data efficiently [5]. Graph Convolutional Networks (GCNs) are an important variant of GNNs. Through multi-layer convolutional operations, GCNs can gradually extract and integrate the local and global features of the nodes while preserving the structural information of the graph. This process achieves the learning and predictive representation of the entire graph [6].

Daniel et al. (2018) first proposed an adversarial attack model on attribute graphs and investigated challenging poisoning and causal attacks [7]. In the same year, Chandler Zuo explored the regularization effect of the Fast Gradient Sign Method (FGSM) based on IG-FGSM within the framework of the Generalized Linear Model (GLM). The IG-FGSM algorithm model was proposed based on this work [8]. Thomas et al. proposed a scalable semi-supervised learning method based on graph convolutional networks [9]. Joan et al. explored the possibility of performing convolutional operations on signals defined in more general domains without the need for translation groups [10].

B. Hypergraph Learning

Hypergraphs, as a generalization of graphs, have significant advantages [11] over simple graph structures in characterizing complex relationships and expressing higher-order features. Combining hypergraphs with neural network models can enhance the ability to manage relationships between data. William et al. (2017) proposed an inductive representation learning method, Graph SAGE, which samples features of neighboring nodes and aggregates them into central node features [12]. In the same year, Cao et al. proposed a new hypergraph neural network model, HGNN, which effectively handles higher-order relationships and better captures complex dependencies between nodes [13]. Building on the hypergraph neural network model, Jiang et al. proposed the dynamic hypergraph construction (DHGCN) method [14]. This method divides the model into two parts, enhancing the accuracy of the hypergraph structure through network construction and updating.

III. Method

In this section, we will introduce the proposed improved probability distribution hypergraph neural network model in detail. The network framework is shown in Fig. 3.1. The model consists of two parts: probability distribution hypergraph construction and adaptive hypergraph convolution. We will first introduce these two parts.

A. Probability distribution hypergraph construction

Given a feature embedding set $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, where $\mathbf{x}_i (i = 1, 2, \ldots, n)$ denotes the feature of the ith sample. In a hypergraph, vertex u denotes a sample, and hyperedge e denotes a collection of samples. Thus, a hypergraph can be expressed as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the vertex set and $\mathcal{E}$ denotes the set of hyperedges.

We introduce a probability distribution space to represent the multidimensional label distribution, characterizing the implied object properties. The feature similarity of the object is calculated using the Euclidean distance formula as follows:

$$r_{ic} = \frac{\pi_c N\left(x_i \mid \mu_c, \Sigma_c\right)}{\sum_{k=1}^{K} \pi_k N\left(x_i \mid \mu_k, \Sigma_k\right)} \tag{1}$$

Here, the specific feature variables of the ith and jth samples in the matrix $\mathbf{X}$ are represented in the rth sample feature variable of the ith sample, and R represents the total number of features.

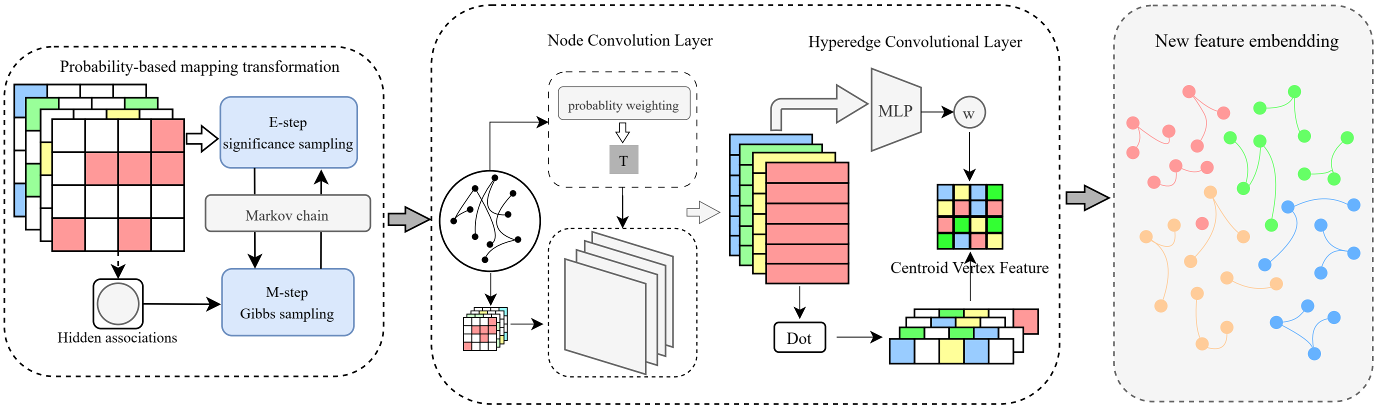


Fig. 2. Overview of Our DHGNN Framework. The first part of the framework describes the process of hypergraph construction for the central vertex and its neighborhood. It uses a GMM algorithm, improved by the sampling method, to iteratively generate object-label mapping relationships. This method incorporates implicit relationships into the node features and generates the hypergraph structure based on probability distributions. In the second part, the vertex features in the hypergraph are aggregated into one-dimensional hypergraph features through vertex convolution. Meanwhile, the features in the neighboring hyperedges are aggregated into central vertex features through hyperedge convolution. After performing these operations on all vertices of the current layer feature embedding, we obtain a new feature embedding. This new embedding will be used to construct a new hypergraph structure, as shown in the third frame.

Next, we use a hybrid Gaussian model optimized for the sampling method to compute a matrix of object feature probability distributions for each label cluster. This model is trained using a modified Expectation Maximisation (EM) algorithm [15]. Using importance sampling to approximate the expectation computation in the E-Step reduces the computational complexity of the model. Due to the complexity of the original target probability distribution, we use the normal distribution to approximate it. The specific sampling formula is as follows:

$$E(f) = \int f(x)\frac{p(x)}{q(x)}q(x)\,dx \approx \frac{1}{N}\sum_{n=1}^{N}\frac{p(x)}{q(x)}f(x) \tag{2}$$
$$x^n \sim q(z)$$

where the importance weight is $w^n = \frac{p(x^n)}{q(x^n)}$. At this point, a weighted importance sampling method is used to minimize the difference between the approximate probability distribution and the original probability distribution:

$$E(f) \approx \sum_{n=1}^{N}\frac{w^n}{\sum_{n=1}^{N} w^m}f(x^n) \tag{3}$$

The distribution of importance weights uses a normal distribution as an approximation. At each iteration, the probability distribution obtained from the previous sample is used as a trajectory to compute the importance weights for the current iteration.

To compute the implied parameters in the probability distributions, our maximization formula for θ in the M-step can be converted into a full log-likelihood expectation calculation using the following formula:

$$Q(\theta, \theta^{old}) = \int p(Z|X, \theta^{old})\ln p(Z|X,\theta)\,dZ \tag{4}$$

where Z is sampled using specific samples, and the computation of $p(Z|X,\theta)$ is as follows:

$$p(Z|X,\theta) = \frac{Z^x(1-Z)^{1-x}\cdot\frac{\alpha}{\sqrt{2\pi}\sigma}e^{-\frac{(Z-1/2)^2}{2\sigma^2}}}{\int_0^1 Z^x(1-Z)^{1-x}\cdot\frac{\alpha}{\sqrt{2\pi}\sigma}e^{-\frac{(Z-1/2)^2}{2\sigma^2}}\,dZ} \tag{5}$$

Here, the Markov Chain Monte Carlo (MCMC) method is used to sample the new round of iterations using the old values. This sampling method effectively samples the multi-dimensional target distribution in a multivariate environment. It circumvents the drawback of not being able to directly calculate the posterior distribution. Additionally, using multiple iterations helps reduce the errors generated by sampling.

In the first iteration of hidden variables, a random sample is obtained as $\Theta = (\theta_1^1, \theta_2^1, \ldots, \theta_n^1)$, where the joint probability distribution $P(\theta_1^1|\theta_2^1, x_1, x_2, x_3, \ldots, x_n)$ is used. We simplify the above formula by converting the joint posterior distribution into a conditional probability formula:

$$\begin{aligned} &p(\theta_1^1|\theta_2^1, x_1, x_2, x_3, \ldots, x_n) \\ &\quad\propto p(\theta_1^1, \theta_2^1|x_1, x_2, x_3, \ldots, x_n) \\ &\quad\propto p(x_1, x_2, x_3, \ldots, x_n|\theta_1^1, \theta_2^1)p(\theta_1^1, \theta_2^1) \end{aligned} \tag{6}$$

Therefore, the Q function in the improved M-step can be expressed as:

$$Q(\theta, \theta^{old}) =$$
$$\int p(\theta_1, \theta_2, \ldots, \theta_n, \theta^{old})\ln p(\theta_1, \theta_2, \ldots, \theta_n, \theta)\,d\theta_1 d\theta_2 \ldots d\theta_n \tag{7}$$

On this basis, we iteratively train using the standard EM algorithm to obtain the object's probability distribution matrix. Finally, the probability distribution matrix and the hypergraph Laplacian matrix are used as inputs to generate the label cluster-based probability distribution hypergraph.

B. Adaptive hypergraph convolution

The adaptive hypergraph convolution method consists of two parts: the adaptive vertex convolution module and the hyperedge convolution module. The adaptive vertex convolution module creates a transformation matrix through multidimensional probability distributions to induce vertex features to approximate high-likelihood label clusters and form hyperedge features. It then continues to convolve the hyperedge features to aggregate hyperedges with similar probability distributions near the central vertex.

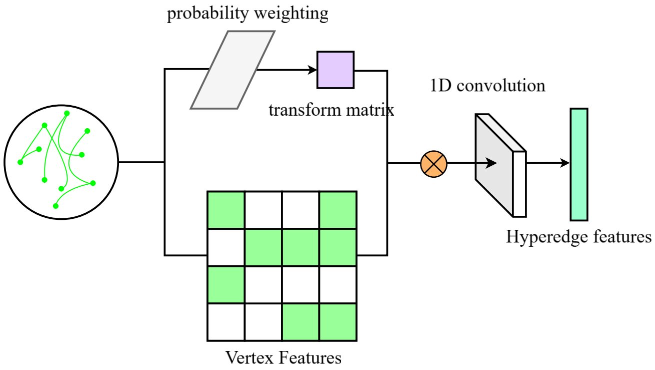


Fig. 3. Adaptive Vertex Convolution Module. For k vertices, the $k \times k$ transformation matrix is calculated by combining the probability distribution matrices of the vertices. After being processed by the multilayer perceptron, the transformed vertex feature matrix is multiplied by the transformation matrix to obtain the transposed and weighted vertex feature matrix. Then, one-dimensional convolution is performed to obtain the one-dimensional hyperedge features.

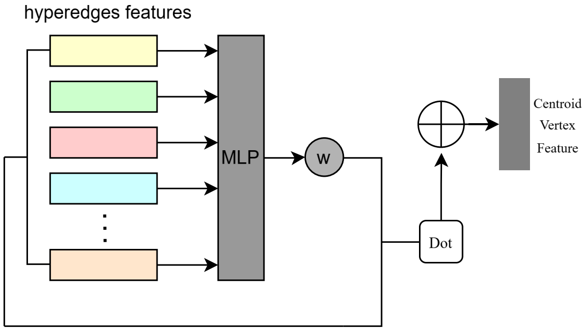


Fig. 4. The hyperedge features are aggregated using a self-attentive mechanism. The adaptive weights w are computed from the hyperedge features through an MLP. The average of the adaptive weights of the hyperedge features is then computed as the central vertex feature.

1) Adaptive vertex convolution: Most popular vertex convolution methods today use pre-computed transformation matrices to perform convolution operations on vertices, which are an extension of pooling schemes. These methods are better suited for explicit object-label mapping relationships that do not contain implicit attributes. To better adapt to these implicit relationships and one-to-many object-label mappings, we propose learning the transformation matrix through the multidimensional probability distributions of the object-label clusters. This matrix is used to weight and rank the vertices, as shown in Figure 3.

During the convolution process, we use a multilayer perceptron (MLP) combined with conditional probability to generate the transformation matrix T_p. The transformed features are then compressed using 1-D convolution [16]. The formula is as follows:

$$\begin{aligned} \mathrm{T}_p &= MLP(p(k)\mathbf{X}_u) \\ p(k) &= \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \end{aligned} \tag{8}$$

$$\mathbf{x}_e = \mathrm{conv}(\mathbf{T} \cdot MLP(\mathbf{X}_u)) \tag{9}$$

2) Hyperedge convolution: As shown in Fig. 4, the hyperedge convolution module aggregates one-dimensional hyperedge features into central vertex features, thereby converting one-to-many object-label mappings into one-to-one mappings. Specifically, the weight scores of the hyperedge features are generated using an MLP. The one-to-one central vertex features are then obtained by performing weighted averaging based on these weight scores. The formula is as follows:

$$\begin{aligned} \omega &= \mathrm{softmax}(b + \sum x_e * P_e) \\ x_u &= \sum_{i=0}^{|Adj_u|} \omega^i x_e^i \end{aligned} \tag{10}$$

IV. Applications and Experiments

In this section, we apply our model to citation networks with intrinsic graph structures. Specifically, for data with intrinsic graph structures, we sample k vertices in the 1st-order neighborhood of u. These k vertices also form a hyperedge in Adj(u).

A. Experiments in categorizing citation datasets

An open-source text dataset is selected for experimental validation of the theoretical methods proposed in this chapter. Controlled experiments are conducted on the textual dataset to compare the improved GMM hypergraph construction method and the probabilistic hypergraph convolution module proposed in this chapter against the conventional hypergraph convolution model. The effectiveness and superiority of the two approaches are verified on the textual dataset and data containing hidden high-latitude relations through different ablation experiments.

1) Cora dataset: The Cora dataset is used for categorizing scholarly literature, mainly in machine learning and natural language processing research. The nodes in the knowledge network represent papers, and the edges connecting the nodes represent citation relations. Each node includes information such as title, author, and more. Based on this information, the dataset categorizes papers into one or more categories, such as artificial intelligence, data mining, and target recognition research.

2) Experiment setup: We set specific experimental metrics for the label classification task on the Cora dataset. For the citation label classification task, we use test metrics including Accuracy (ACC) and Loss Rate (LOSS), while we define LR as the training set size of the dataset. The experiments compare our method with GCN model, HGCN model and DHGCN model based on K-means method and the results are presented in graphical form.

3) Result: We compared our method with the latest graph/hypergraph-based neural network methods on the Cora dataset. The experimental results are shown in Table 1, where our method outperforms state-of-the-art methods by 0.6% to 2% when using 1% to 8% randomly sampled data as the training set. The results show that the hypergraph structure is more competitive compared to the graph structure when the dataset is small. Additionally, the introduction of the probability distribution enhances the hypergraph structure's ability to express implicit relationships.

TABLE I
PERFORMANCE COMPARISON USING DIFFERENT METHODS ON CORA

LR	#train	HGCN	DHGCN	DHGCN(ours)
1	27	74.8	79.4	80.8
2	54	75.4	80.3	80.9
4	108	78.8	80.7	81.7
6	162	79.0	80.9	82.9
8	216	79.6	80.9	82.6
10	270	79.9	81.6	81.3

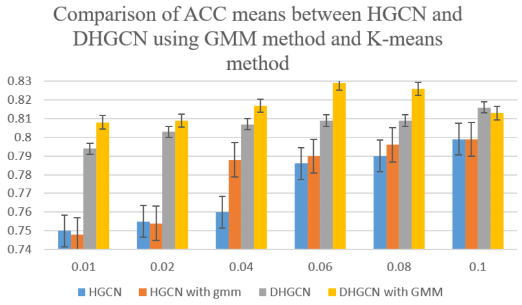


Fig. 5. Comparison of ACC means between HGCN and DHGCN using GMM method and K-means method.

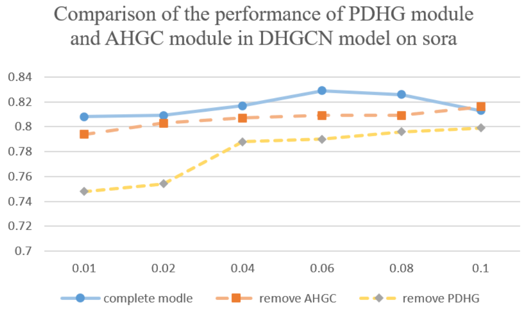


Fig. 6. Performance comparison of PDHG and AHGC modules in DHGCN model on sora dataset.

4) Ablation experiments: To demonstrate our method's effectiveness, this paper conducts multiple sets of ablation experiments under the same experimental conditions. The ablation experiments compare the performance of the hypergraph representation using the K-means method with that of the DHGCN model, both with and without the PDHG and AHGC modules. Our method is also evaluated under the same experimental conditions. In the case where the AHGC module is removed, the model employs the standard convolutional module on the Cora dataset. The results of the ablation experiments using the K-means method are presented in Fig. 5, while the results of experiments with the AHGC and PDHG modules removed are depicted in Fig. 6. The experimental data demonstrate that our method achieves superior mapping results compared to the traditional K-means method. This improvement leads to a 2% increase in ACC over the traditional DHGCN model. Compared to the traditional HGCN model, the DHGCN model incorporating the AHGC module demonstrates faster convergence and improved label classification accuracy. The inclusion of the AHGC module leads to a 4% enhancement in label classification accuracy. Furthermore, when the dataset size reaches 6%, the model's performance peaks with a 3% improvement.

B. Experiments in Sentiment Analysis of Multi-modal Datasets

In addition to the citation network experiments, we evaluated our model on a more complex task, namely sentiment analysis on multi-modal datasets. Multi-modality is an important feature of social media. We use hypergraphs to model higher-order relationships between different modalities. The multi-modal hypernodes jointly represent the correlations between vertices. In our experiments, we use the MELD dataset to evaluate our hypergraph model.

1) MELD dataset: MELD (Multi-modal Emotion Line Dataset) is a multi-modal emotion recognition dataset derived from the classic English TV series ”Friends”. The MELD dataset consists of 1,400 dialogue snippets with a total of 13,000 segments. Each dialogue segment is labeled with seven emotions including anger, disgust, sadness, joy, neutrality, surprise, and fear.

2) Experiment setup: We constructed three hyperedge sets for each of the three modalities and merged these sets into one multi-modal hyperedge set. The number of vertices contained in each cluster is 64. We select 2 nearest clusters and a labeled cluster from the GMM clustering as the neighboring hyperedge set for each vertex. We compare our model with recent multi-modal sentiment prediction methods, Dialogue RNN method [17]; Dialogue GCN method [18]; MMGCN method [19].

3) Result: The experimental results of the different methods on the MELD dataset are shown in Table 2, where each model obtains an average F1 weight score on the MELD dataset. It can be seen that our method improves the performance by 7% to 8% over the Dialogue RNN and Dialogue GCN methods, and by 5% compared to MMGCN.

4) Ablation experiments: To demonstrate the state-of-the-art performance of our method, we conducted a series of

TABLE II
PERFORMANCE COMPARISON USING DIFFERENT METHODS ON MELD

Method	ACC
Dialogue RNN	57.11
Dialogue GCN	58.23
MMGCN	60.84
our method	65.12

ablation experiments using the K-means clustering method and compared the performance of our clustering method on the dataset, as shown in Fig. 7. The experimental data demonstrate that our method achieves superior mapping results compared to the traditional K-means method, with an experimental ACC that is 4% better than the traditional DHGCN model. Furthermore, the model performance reaches its optimal level when the number of convolutional layers reaches 8.

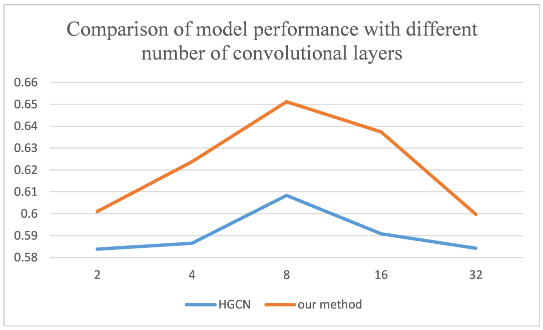


Fig. 7. Comparison of ACC means between HGCN and DHGCN using GMM method and K-means method.

V. CONCLUSIONS

In this study, we propose enhancements to the hybrid Gaussian model for multi-mapping sample classification. Our improvements involve replacing the uniform mapping space with probability distributions for hypergraph representation and optimizing the hypergraph structure for input data features. We also introduce a dynamic hypergraph convolution module based on multidimensional probability distributions to dynamically prune and update the structure, remove erroneous associations and low-confidence relationships, and express high-latitude associations among the data, while also uncovering logical relationships between hidden features.

After thoroughly validating the reliability and effectiveness of the model using the Sora dataset, we found that our approach achieves similar or superior performance compared to state-of-the-art methods. Additionally, we have investigated the validity of the PDHG module and the AHGC module.

In future work, it may be beneficial to explore improved and more interpretable hypergraph construction methods.

REFERENCES

[1] Li, Z., Liu, F., Yang, W., et al. ”A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects.” *IEEE Trans Neural Netw Learn Syst*, vol. 33, no. 12, pp. 6999-7019, 2022.

[2] Zhang, Z., Cui, P., Zhu, W. ”Deep Learning on Graphs: A Survey.” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249-270, 2022.

[3] Feng, Y., Haoxuan, Y., Zizhao, Z., et al. ”Hypergraph Neural Networks.” 2018.

[4] Jiang, J., Yuxuan, W., Yifan, F., et al. ”Dynamic Hypergraph Neural Networks.” *PROCEEDINGS OF THE TWENTY-EIGHTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pp. 2635-2641, 2019.

[5] Zhou, J., Ganqu, C., Shengding, H., et al. ”Graph neural networks: A review of methods and applications.” *AI Open*, pp. 157-81, 2020.

[6] Li, J., Tao, X., Liang, C., et al. ”Adversarial Attack on Large Scale Graph.” 2020.

[7] Alom, M. Z., Paheding, S., Tarek-M, T., et al. ”Handwritten Bangla Digit Recognition Using Deep Learning.” 2017.

[8] Zuo, C. ”Regularization Effect of Fast Gradient Sign Method and its Generalization.” Ithaca: Cornell University Library, arXiv.org, 2018.

[9] Li, J., Yongfeng, H., Heng, C., et al. ”Semi-Supervised Hierarchical Graph Classification.” 2022.

[10] Bruna, J., Mallat, S. ”Invariant scattering convolution networks.” *IEEE Trans Pattern Anal Mach Intell*, vol. 35, no. 8, pp. 1872-1886, 2013.

[11] Ramos, M. V. M., Queiroz, R. J. G. B. ”Formalization of simplification for context-free grammars.” 2015.

[12] Ahmed, N. K., Rossi, R. A., Zhou, R., et al. ”Inductive Representation Learning in Large Attributed Graphs.” 2017.

[13] Zhang, H., Xin, L., Lidong, B. ”Video-LLaMA: An Instruction-tuned Audio-Visual Language Model for Video Understanding.” arXiv.org, 2023.

[14] Liu, L., Feng, H., Xuan, L., et al. ”Multi-view Contrastive Learning Hypergraph Neural Network for Drug-Microbe-Disease Association Prediction.” In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 4829-4837, Macao, China, 2023.

[15] Ramos, M. V. M., Queiroz, R. J. G. B. ”Formalization of simplification for context-free grammars.” 2015.

[16] Zaremba, W., Sutskever, I., Vinyals, O. ”Recurrent Neural Network Regularization.” 2014.

[17] Majumder, N., Poria, S., Hazarika, D., et al. ”DialogueRNN: An Attentive RNN for Emotion Detection in Conversations.” Honolulu: 2019, pp. 6818-6825.

[18] Ghosal, D., Majumder, N., Poria, S., et al. ”DialogueGCN: A graph convolutional neural network for emotion recognition in conversation.” arXiv, 2019.

[19] Hu, J., Yuchen, L., Jinming, Z., et al. ”MMGCN: Multimodal Fusion via Deep Graph Convolution Network for Emotion Recognition in Conversation.” Ithaca: Cornell University Library, arXiv.org, 2021.