

JVLC

**Journal of
Visual Language and
Computing**

Volume 2024

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/JVLC2024

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

**Journal of
Visual Language and Computing**

Editor-in-Chief

Shi-Kuo Chang, University of Pittsburgh, USA

Co-Editors-in-Chief

Gennaro Costagliola, University of Salerno, Italy

Paolo Nesi, University of Florence, Italy

Franklyn Turbak, Wellesley College, USA

An Open Access Journal published by

KSI Research Inc.

156 Park Square Lane, Pittsburgh, PA 15238 USA

JVLC Editorial Board

Tim Arndt, Cleveland State University, USA

Paolo Bottoni, University of Rome, Italy

Stefano Cirillo, University of Salerno, Italy

Francesco Colace, University of Salerno, Italy

Nathan Eloe, University of North Western Missouri, USA

Martin Erwig, Oregon State University, USA

Andrew Fish, University of Brighton, United Kingdom

Vittorio Fucella, University of Salerno, Italy

Angela Guercio, Kent State University, USA

Jun Kong, North Dakota State University, USA

Robert Laurini, University of Lyon, France

Mark Minas, University of Munich, Germany

Brad A. Myers, Carnegie Mellon University, USA

Kazuhiro Ogata, JAIST, Japan

Genny Tortora, University of Salerno, Italy

Kang Zhang, University of Texas at Dallas, USA

Yang Zou, Hohai University, China

Journal of Visual Language and Computing

Volume 2024

December 2024

Table of Contents

Regular Papers

Accurate Recognition of Drill Core Images with an Improved ResNet Model.....1 <i>Yong Pu, Zhihao Shao, Ziyuan Xu, Chuanhu Xiong and Yonghua Chen</i>	1
Enhancing API Retrieval through Multi-Source information Knowledge Graph Construction. 11 <i>Weiwei Wang, Zijie Che, Ruilian Zhao, Zhan Ma and Ying Shang</i>	11
MVKGCL: Recommendation Model Based on Knowledge Graph and Comparative Learning..... 23 <i>Zhiyue Xiong, Hankiz Yilahun, Sadiyagul Anwer and Askar Hamdulla</i>	23
An Improved DBNet Model with Pyramid Pooling and Multi-Scale Enhancement for Small Text Object Detection. 31 <i>Zhu Yuan, Wuxuan Tang, Jin Yao and Chengjun Liu</i>	31
Educational Video Ecosystems: An Interactive Mind Map-Driven Approach 41 <i>Taeghyun Kanga, Hyungbae Park and Sunae Shin</i>	41

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

Accurate Recognition of Drill Core Images with an Improved ResNet Model

Yong Pu^a, Zhihao Shao^b, Ziyuan Xu^{b,*}, Chuanhu Xiong^a and Yonghua Chen^a

^a Guangzhou Metro Design & Research Institute Co., Ltd., Guangzhou, China

^b Institute of Intelligence Science and Technology, School of Computer Science and Software Engineering, Hohai University, Nanjing, China

ARTICLE INFO

Article History:

Submitted 8.19.2024

Revised 8.31.2024

Second Revision 10.31.2024

Accepted 11.15.2024

Keywords:

Drill core image recognition

Self-calibration

Three-dimensional attention

MAM pooling

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, MSL-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.

© 2024 KSI Research

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

*Corresponding author

Email address: hideon27@hhu.edu.cn

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 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 MSL-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 MSL-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, MSL-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.

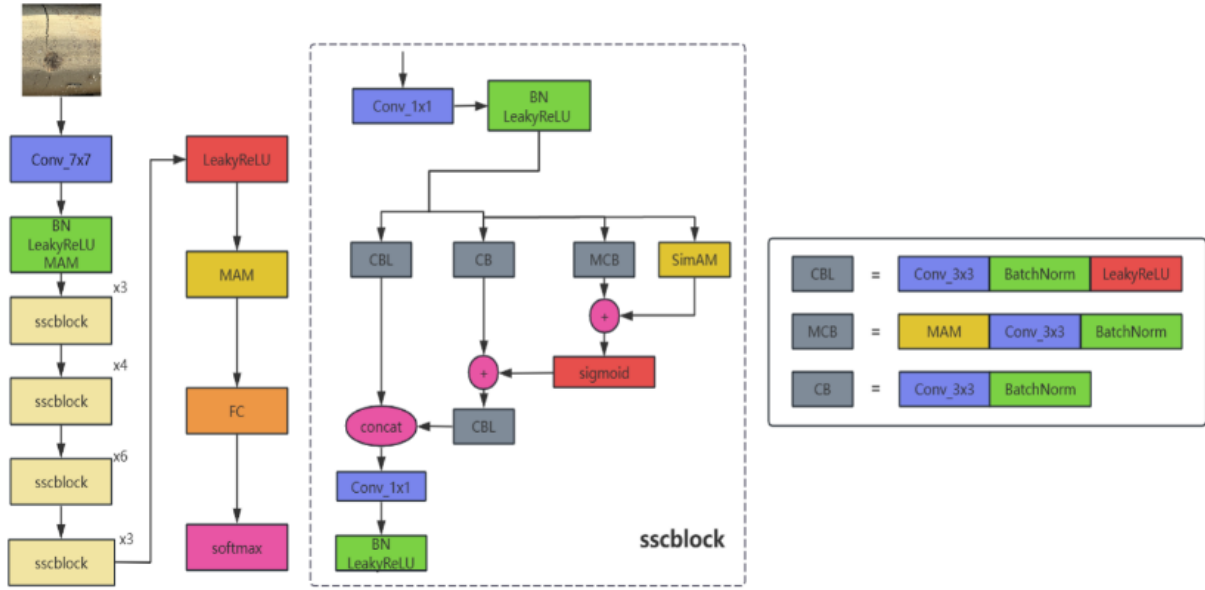


Figure 1: The structure of the proposed model MSL-ResNet50.

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 $K = [k_1, k_2, \dots, k_c]$, and input data $X = [x_1, x_2, \dots, x_c]$, the output data after convolution is denoted as $Y = [y_1, y_2, \dots, y_c]$. The output of the i -th channel can be expressed as $y_i = k_i * X = \sum_{j=1}^c k_i^j * x_j$.

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.

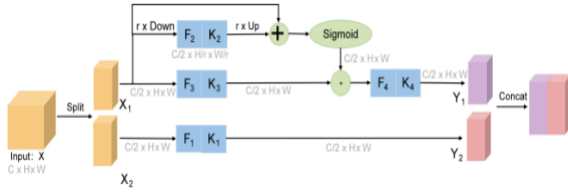


Figure 2: The structure of self-calibrated convolution module.

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:

$$T_1 = AvgPool_r(X_1) \quad (1)$$

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

$$X'_1 = Up(F_2(T_1)) = Up(T_1 * K_2) \quad (2)$$

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

$$M_1 = \sigma(X_1 + X'_1) \quad (3)$$

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 :

$$\begin{cases} Y'_1 = F_3(X_1) \cdot M_1 = (X_1 * K_3) \cdot M_1 \\ Y_1 = F_4(Y'_1) = K_4 * Y'_1 \end{cases} \quad (4)$$

where $F_3(X_1) = X_1 * K_3$, and \cdot 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.

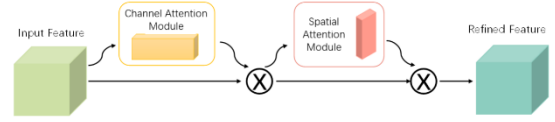


Figure 3: The CBAM structure.

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.

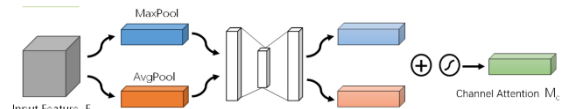


Figure 4: The channel attention structure.

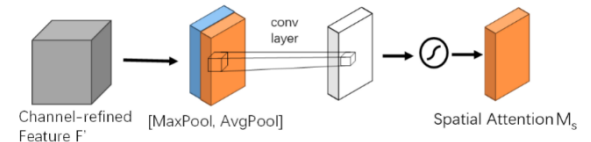


Figure 5: The spatial attention structure.

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:

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

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 λ is added to minimize Equation 5, yielding the final energy function:

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

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:

$$\begin{cases} w_t = -\frac{2(t - \mu_t)}{(t - \mu_t)^2 + 2\sigma_t^2 + 2\lambda} \\ b_t = -\frac{1}{2}(t + \mu_t)w_t \\ \mu_t = \frac{1}{M-1} \sum_{i=1}^{M-1} X_i \\ \sigma_t^2 = \frac{1}{M-1} \sum_{i=1}^{M-1} (x_i - \mu_t)^2 \end{cases} \quad (7)$$

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:

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

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 \tilde{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 .

$$\tilde{X} = \text{sigmoid}\left(\frac{1}{E}\right) \odot X \quad (9)$$

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.

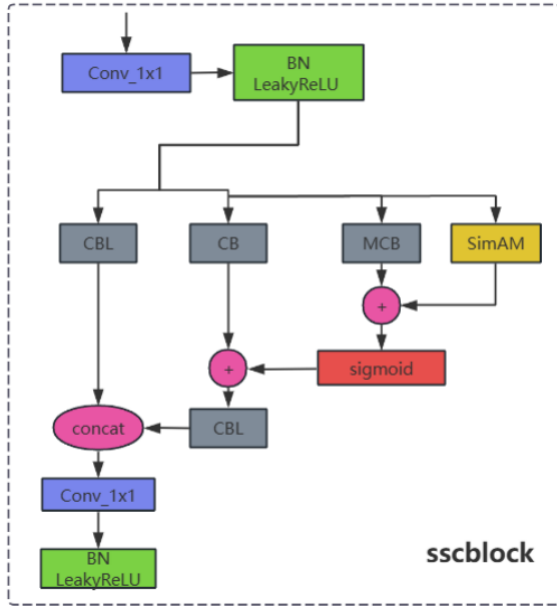


Figure 6: The structure of self-calibration residual with the SimAM attention.

4. MAM Pooling

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

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

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, α and β 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.

(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.

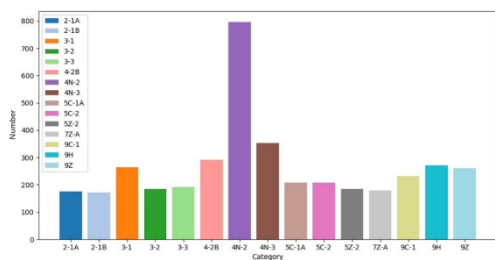


Figure 7: Number of samples in each category.

(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.



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.

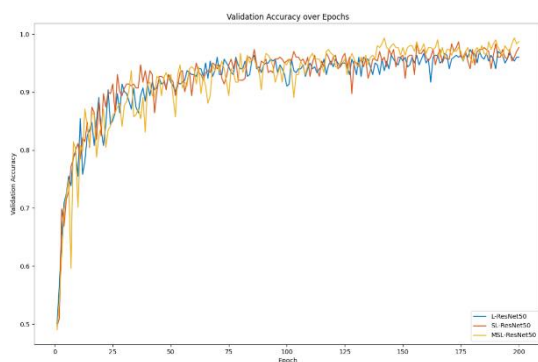


Figure 10: The curves of validation accuracy of the three ResNet50 models.

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.

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

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

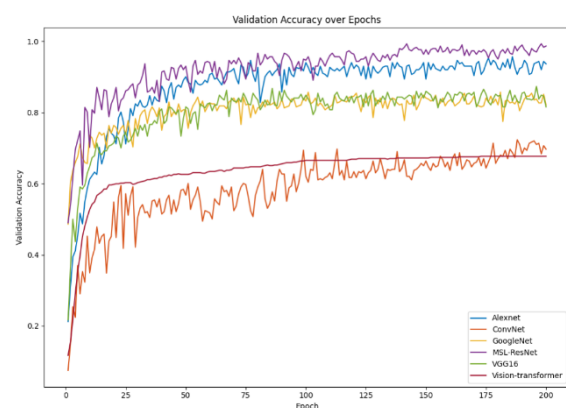


Figure 11: The curves of validation accuracy of the involved models for comparison.

(5) Grad-CAM feature visualization

Grad-CAM was employed to generate heatmaps for some selected samples, as shown in Figure 12. It can be observed that certain regions in the image samples

receive high attention from the model, such as texture, color variations, and morphological features. These regions may contain features crucial for the model's classification decisions. Notably, these high-attention areas are concentrated in the actual core sections of the images rather than the background, indicating that the proposed model's focus on the target object' regions is appropriate and captures useful geological features.

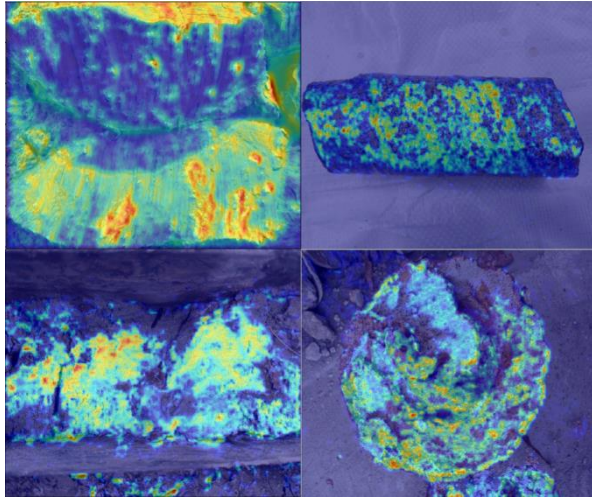


Figure 12: The curves of validation accuracy of the involved models for comparison.

6. Conclusion

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. Alferez, E. L. Vazquez, 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.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc

Enhancing API Retrieval through Multi-Source information Knowledge Graph Construction

Weiwei Wang^a, Zijie Che^b, Ruilian Zhao^b, Zhan Ma^b and Ying Shang^{b,*}

^aCollege of Information Engineering, Beijing Institute of Petrochemical Technology, Beijing, P.R.China

^bCollege of Information Science and Technology, Beijing University of Chemical Technology, Beijing, P.R.China

ARTICLE INFO

Article History:

Submitted 7.20.2024

Revised 8.31.2024

Second Revision 9.28.2024

Accepted 10.30.2024

Keywords:

API knowledge graph

Multi-source fusion

Knowledge extraction

API recommendation

API retrieval

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.

© 2024 KSI Research


1. 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[4]. 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[25], BIKER[4], RACK [14] 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

 wangweiwei@bipt.edu.cn (W. Wang); 747938871@qq.com (Z. Che); r1zhao@mail.buct.edu.cn (R. Zhao); shangy@mail.buct.edu.cn (Y. Shang)

effectively represent the semantic association between information, which is suitable for expressing API-related knowledge. For example, Liu et al.[10] 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.[9] 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[19]. 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[13]. 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 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 an-

swer 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.

2. Backgrounds

We briefly describe PyTorch-related APIs, API documentation and APIs from Stack Overflow for readers to better follow the content of the paper

2.1. PyTorch-related APIs

PyTorch is an open-source machine learning library that is primarily used for building deep neural networks. It was developed by Facebook’s AI research team and has become a popular choice among researchers and developers due to its ease of use and flexibility. The official PyTorch API documentation is abundant and specific. Further, there are many Q&A related to PyTorch in *SO*. Thus, this paper focuses on PyTorch-related APIs.

2.2. APIs Documentation

API documentation refers to the document that provides information on how to use a particular API. It generally contains the functional description and the structure of the API, including the functions, methods, parameters, and endpoints of an API, as well as any relevant code examples and usage guidelines. This documentation is used by developers who want to integrate the API into their software or applications, and it helps them understand how to make requests to the API and receive responses. The API documentation is the most effective resource for extracting the knowledge of APIs.

2.3. APIs from Stack Overflow

Stack Overflow is a question and answer community website for programmers. It was launched in 2008 and has since become a popular platform for developers to ask and answer technical questions related to programming. It starts from users’ practical problems and obtains comprehensive answers for different application scenarios. The API-related Q&A information aims to solve practical problems, which is not available in API documentation. So, in recent years, many researchers have used Stack Overflow to make up for the shortcomings of API documentation. The *SO* website has become another important resource for developers to learn about the knowledge of APIs.

2.4. Entity Extraction Techniques

Entity extraction refers to the process of extracting key information from data sources that are not uniformly structured and converting them into structured data through relevant techniques. In the field of software engineering, there are several approaches based on different principles of entity extraction.

2.4.1. Rule-based entity extraction

Ye et al.[23] proposed a rule-based entity extraction method, which mostly uses keywords, central words, superlatives, subordinate words, punctuation marks and other features in the text. And these features are combined to construct rules. With these rules, pattern matching and string matching can be used to complete entity extraction. This approach relies on the creation of a complete knowledge base and lexicon.

2.4.2. Template-based entity extraction

Stephen et al.[16] proposed an approach for entity extraction through natural language processing (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.[8] 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.

2.4.3. Statistical-based entity extraction

Statistical-based methods mainly use maximum entropy, support vector machines, and conditional random fields to train feature sets selected from texts as a way to automatically label and extract entity types[3]. Ye et al.[22] proposed a semi-supervised entity extraction method for software engineering forum content, which is oriented to Stack Overflow, labeled with different types of software-specific entities and used CRFs to statistically model their sequential data annotation, and finally designed a software-specific entity extraction system. This method not only solves the challenge of defining comprehensive rules in rule-based methods, but also avoids the problem of unreliable dictionary matching.

3. 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.

3.1. Knowledge Acquisition of 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.

3.1.1. Knowledge representation

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.

3.1.2. Knowledge extraction

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[20] 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 "*standard deviation*" is identified, a "refer to" relationship can be established between the API "*torch.normal*"

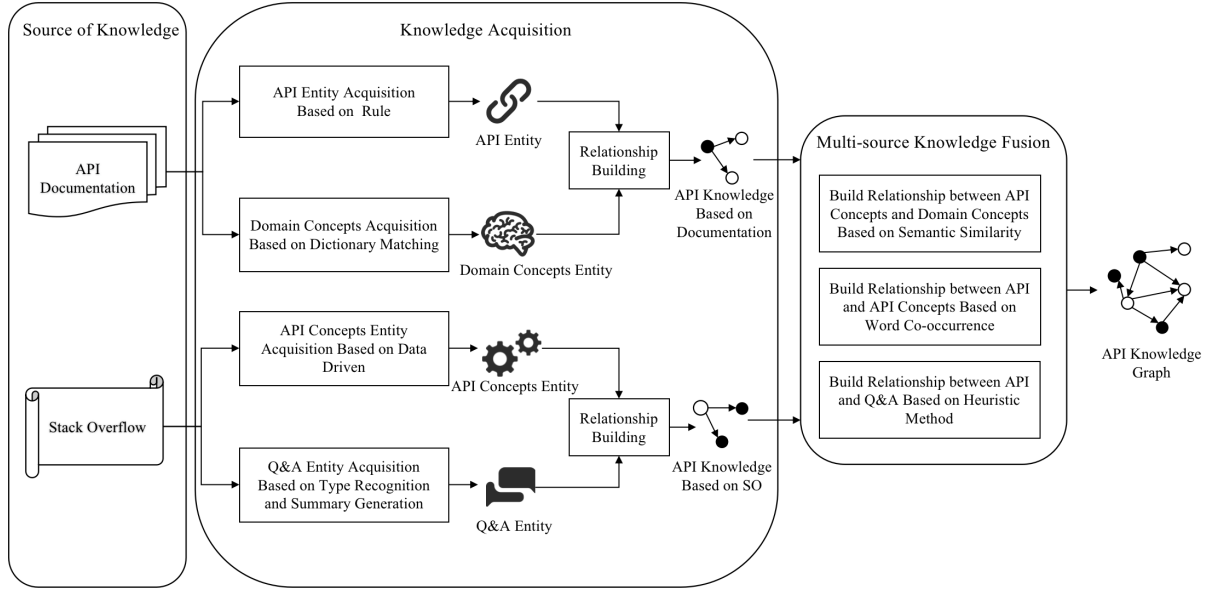


Figure 1: The Framework of API Knowledge Graph Based on Multi-Source Information Fusion

and the domain concept "standard deviation".

3.2. 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.

3.2.1. Knowledge Representation

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[13], 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.

3.2.2. Knowledge extraction

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.

Table 1
Semantic template of indicating prominent Paragraphs

Number	Sentence	Number	Sentence	Number	Sentence
1	Please check XX	9	I'd recommend XX	17	If you want to XX
2	Pls check XX	10	In summary, XX	18	Simply out XX
3	You should XX	11	Keep in mind that XX	19	If you use XX
4	You can try XX	12	I suggest that XX	20	What's important XX
5	You could try XX	13	You can use XX	21	When you call XX
6	Check out XX	14	I prefer XX	22	By popular demand XX
7	In short, XX	15	XX needs to be called	23	You would need to XX
8	The most important is XX	16	Note that XX	24	You must XX

In more detail, based on the categorization of *SO* Q&A by Stefanie Beyer et al.'s [1], 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[15]. 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]. For semantic templates, this paper expands the templates proposed by Xu et al.[21] by combining the sentence structure features of API recommendation and summaries, which are shown in Table 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.

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

(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, a 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.

3.3. 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.

3.3.1. 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 .

$$freq(A_i \rightarrow AC_j) \geq \alpha \quad (3)$$

3.3.2. 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[12] 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.

$$sim_{lex}(n_1, n_2) = \frac{|Token(n_1) \cap Token(n_2)|}{|Token(n_1) \cup Token(n_2)|} \quad (4)$$

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

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

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.

3.3.3. 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.

4. 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?

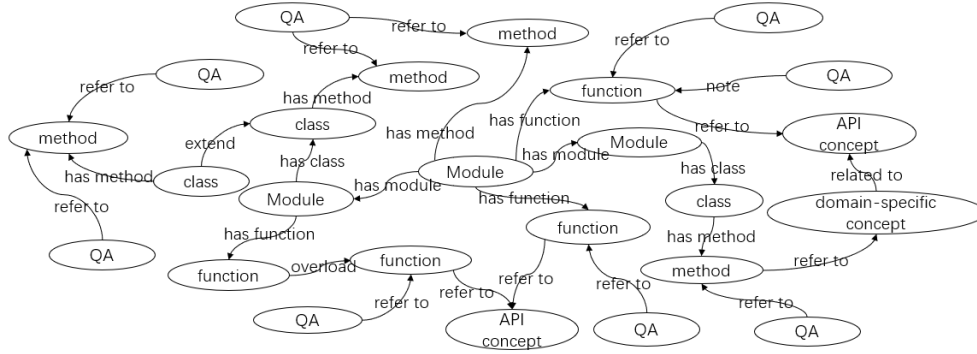


Figure 2: The API knowledge graph based on multi-source information fusion

4.1. 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. Fig.2 is a partial abstract representation of the API knowledge graph.

4.2. 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.

4.3. Experimental Results and Analysis

This part conducts experimental analysis on three research questions to verify the effect of our method.

4.3.1. 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 relation-

Table 2
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

ships. 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), which are commonly used in the literature[7, 6], as comparison methods. The compari-

son of classification effectiveness of different algorithms are shown in Table 3. 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 algorithm is better than other methods.

Table 3
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 4 represents the evaluators' assessment of the 10 *SO* Q&A summaries in Table 2, 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.

4.3.2. 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

Table 4
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

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 5. 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.

4.3.3. 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.

Table 5
Comparison of recommendation results between multi-source information fusion and single-source information

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

These two techniques are used as our comparison methods. The experimental results are shown in Table 6. 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 7 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 6
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 7
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

5. Related work

API as a common component of software development, API retrieval has become a hot topic of current intelligent development research. The knowledge graph provides a solution for the management and efficient retrieval of API knowledge. However, how to build a comprehensive and accurate API knowledge graph with limited resources is still a hot topic of current research. Currently, API recommendation systems are mostly used to solve the problem of API retrieval. Due to the structured feature of knowledge graph, it can represent the knowledge from multiple sources through relationship, and this form of data organization can improve the shortcomings of traditional API recommendation systems and thus enhance the efficiency of API retrieval.

5.1. API Recommendation

Zhang and Jiang et al. [25] proposed an API recommendation system RASH based on API documentation and SO solved historical questions. Ye et al. [24] used word embedding techniques in their study to compensate for the problem of semantically identical but lexical mismatch between query questions and Java API documentation. Based on Zhang and Ye’s research, Huang et al. [4] combined API reference documents, Stack Overflow API corpus and word embedding technology to construct an API recommendation system BIKER with good recommendation effect. These recommendation systems do not use very complex models, but only use the idea of similarity calculation to find similar questions and answers that have been solved in history, and then they can get good recommendation results. However, finding similar questions and answers through similarity calculation in a large number of existing questions and answers cannot guarantee high computational efficiency.

In addition to the above research on recommending APIs based on historically solved problems, existing candidate APIs are selected by lexical similarity measures [2, 11, 18]. Based on the Stack Overflow Q&A corpus, Rahman and Roy [14] first established a mapping database of keywords in questions and APIs in answers, and used two heuristic methods on keyword-API associations to recommend the sorting sequence of APIs for a given query question .

5.2. API Knowledge Graph

The World Wide Web contains a wealth of data information. How to quickly and accurately obtain the content people need from the massive data is a huge challenge for researchers. The biggest difficulty is that most of the information exists in the form of unstructured text, and it is dif-

ficult for people to capture key information from scattered text information. Since Google first proposed the concept of knowledge graph, this structured form of information has provided effective support for efficient knowledge acquisition.

In the existing researches, Li et al. [5] constructed an API warning knowledge graph by extracting warning statements from API documentation and API tutorials to facilitate the retrieval of API warning statements in API reference documents. Sun et al. [17] constructed a task-oriented API knowledge graph to output structured API knowledge; Liu et al. [10] constructed an API knowledge graph by extracting relevant knowledge from API documentation and Wikipedia. Ling et al. [9] constructed an API knowledge graph based on open source projects, and then generate API subgraphs for API-related issues through graph embedding technology. Although the current API knowledge graph construction method can improve the retrieval results of the API, it mainly constructs through information sources such as API documentation, Wikipedia or open source projects. These methods lack information on the description of the actual application scenarios of the API. It is not conducive to the fusion and utilization of API practical application knowledge.

6. 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] Beyer, S., Macho, C., Pinzger, M., Di Penta, M., 2018. Automatically classifying posts into question categories on stack overflow, in: Proceedings of the 26th Conference on Program Comprehension, pp. 211–221.
- [2] Chan, W.K., Cheng, H., Lo, D., 2012. Searching connected api subgraph via text phrases, in: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, pp. 1–11.
- [3] Chen, C., Xing, Z., Wang, X., 2017. Unsupervised software-specific morphological forms inference from informal discussions, in: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), IEEE, pp. 450–461.
- [4] Huang, Q., Xia, X., Xing, Z., Lo, D., Wang, X., 2018. Api method recommendation without worrying about the task-api knowledge gap, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, pp. 293–304.

- [5] Li, H., Li, S., Sun, J., Xing, Z., Peng, X., Liu, M., Zhao, X., 2018a. Improving api caveats accessibility by mining api caveats knowledge graph. in: 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE. pp. 183–193.
- [6] Li, X.X., Li, B., Tian, L.F., Zhang, L., 2018b. Automatic benign and malignant classification of pulmonary nodules in thoracic computed tomography based on rf algorithm. *IET Image Processing* 12, 1253–1264.
- [7] Liang, S., Sabri, A.Q.M., Alnajjar, F., Loo, C.K., 2021. Autism spectrum self-stimulatory behaviors classification using explainable temporal coherency deep features and svm classifier. *IEEE Access* 9, 34264–34275.
- [8] Lin, B., Zampetti, F., Bavota, G., Di Penta, M., Lanza, M., 2019. Pattern-based mining of opinions in q&a websites, in: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE. pp. 548–559.
- [9] Ling, C.Y., Zou, Y.Z., Lin, Z.Q., Xie, B., 2019. Graph embedding based api graph search and recommendation. *Journal of Computer Science and Technology* 34, 993–1006.
- [10] Liu, M., Peng, X., Marcus, A., Xing, Z., Xie, W., Xing, S., Liu, Y., 2019. 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, pp. 120–130.
- [11] McMillan, C., Grechanik, M., Poshyvanyk, D., Xie, Q., Fu, C., 2011. Portfolio: finding relevant functions and their usage, in: Proceedings of the 33rd International Conference on Software Engineering, pp. 111–120.
- [12] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26.
- [13] Nadi, S., Treude, C., 2020. Essential sentences for navigating stack overflow answers, in: 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE. pp. 229–239.
- [14] Rahman, M.M., Roy, C.K., Lo, D., 2016. Rack: Automatic api recommendation using crowdsourced knowledge, in: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), IEEE. pp. 349–359.
- [15] Ryu, S.E., Shin, D.H., Chung, K., 2020. Prediction model of dementia risk based on xgboost using derived variable extraction and hyper parameter optimization. *IEEE Access* 8, 177708–177720.
- [16] Soderland, S., 1999. Learning information extraction rules for semi-structured and free text. *Machine learning* 34, 233–272.
- [17] Sun, J., Xing, Z., Chu, R., Bai, H., Wang, J., Peng, X., 2019. Know-how in programming tasks: From textual tutorials to task-oriented knowledge graph, in: 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE. pp. 257–268.
- [18] Thung, F., Wang, S., Lo, D., Lawall, J., 2013. Automatic recommendation of api methods from feature requests, in: 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE. pp. 290–300.
- [19] Treude, C., Robillard, M.P., 2016. Augmenting api documentation with insights from stack overflow, in: Proceedings of the 38th International Conference on Software Engineering, pp. 392–403.
- [20] Wang, C., Peng, X., Liu, M., Xing, Z., Bai, X., Xie, B., Wang, T., 2019. 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, pp. 97–108.
- [21] Xu, B., Xing, Z., Xia, X., Lo, D.A., 2017. automated generation of answer summary to developers’ technical questions, in: Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, pp. 706–716.
- [22] Ye, D., Xing, Z., Foo, C.Y., Ang, Z.Q., Li, J., Kapre, N., 2016a. Software-specific named entity recognition in software engineering social content, in: 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER), IEEE. pp. 90–101.
- [23] Ye, D., Xing, Z., Li, J., Kapre, N., 2016b. Software-specific part-of-speech tagging: An experimental study on stack overflow, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 1378–1385.
- [24] Ye, X., Shen, H., Ma, X., Bunescu, R., Liu, C., 2016c. From word embeddings to document similarities for improved information retrieval in software engineering, in: Proceedings of the 38th international conference on software engineering, pp. 404–415.
- [25] Zhang, J., Jiang, H., Ren, Z., Chen, X., 2017. Recommending apis for api related questions in stack overflow. *IEEE Access* 6, 6205–6219.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

MVKGCL: Recommendation Model Based on Knowledge Graph and Contrastive Learning

Zhiyue Xiong^a, Hankiz Yilahun^{a,*}, Sadiyagul Anwer^b and Askar Hamdulla^a

^aXinjiang University, Urumqi 830017, China

^bXinjiang Agricultural Vocational Technical University, Urumqi 830017, China

ARTICLE INFO

Article History:

Submitted 11.10.2024

Revised 11.25.2024

Accepted 12.5.2024

Keywords:

Knowledge graphs

Recommendation model

Contrastive learning

Graph attention network

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 the model outperforms others on the Amazon-book and Yelp2018 datasets.

© 2024 KSI Research

1. 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

*Corresponding author

Email address: hansumuruh@xju.edu.cn

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.

2. Related work

2.1 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) 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 an added information aggregation scheme that recursively integrates the relationship paths of remote connections. KGIN provides interpretability for predictions by identifying influential intentions and relationship paths.

2.2 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.

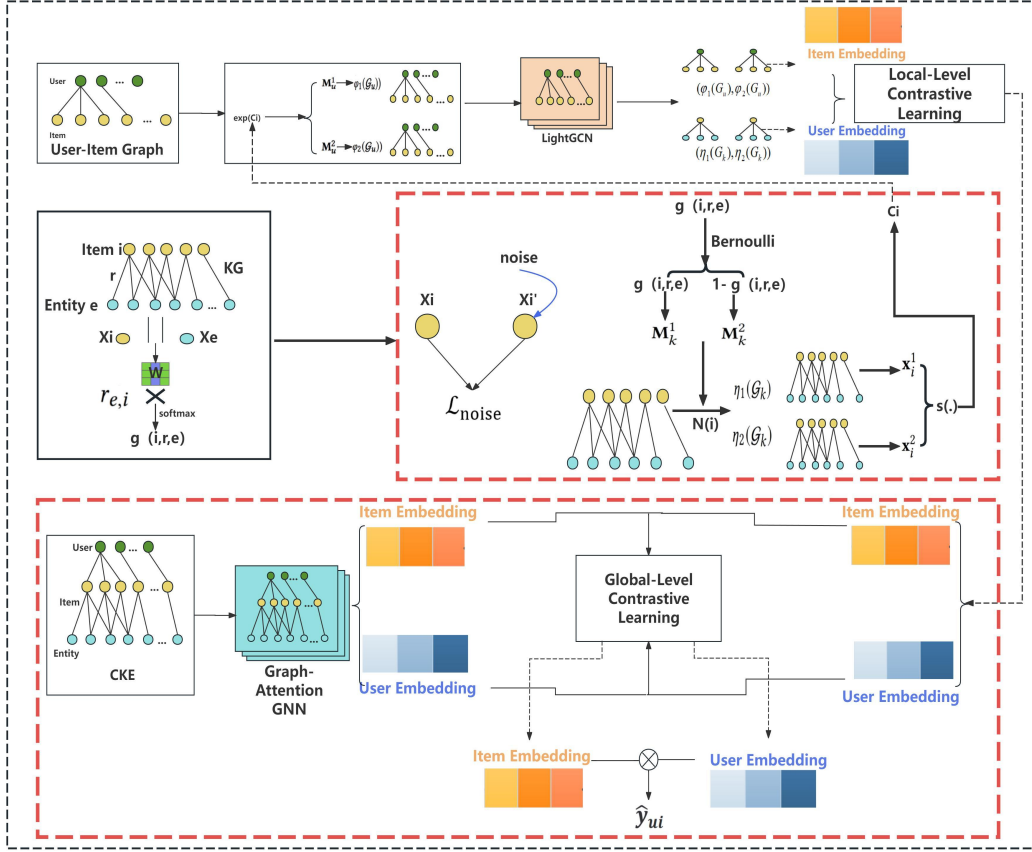


Figure 1: Main characteristics of geospatial rules.

3. Method

3.1 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 $\{(u, Y_{ui}, i) \mid u \in U, i \in I\}$, 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 $A = \{(i, r, e) \mid i \in I, e \in E, r \in R\}$. 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 the 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 $G = \{(h, r, t) \mid h, t \in E', r \in R'\}$, where E' combines entities E from KG with users U ($E' = E \cup U$), and R' expands relations R with Y_{ui} ($R' = R \cup \{Y_{ui}\}$).

3.2 CKG based graph attention network

Firstly, the TransR [23] method is used to generate the embedding representations of CKG. Consider entity h , represented by $N_h = \{(h, r, t) \mid (h, r, t) \in G\}$, 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 .

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

$$\pi(h, r, t) = (W_r e_t)^\top \tanh((W_r e_h + e_r)) \quad (2)$$

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

Following the aggregation of information for entity e_h and its neighborhood combination representation e_{N_h} , we obtain $e_h = f(e_h, e_{N_h})$, where f serves as the aggregator. We further explore 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:

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

$$e_u^{glo} = e_u^{(0)} || \dots || e_u^{(L)}, e_i^{glo} = e_i^{(0)} || \dots || e_i^{(L)} \quad (4)$$

3.3 Automatic masking mechanism

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

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

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

$$g'(e, r_{e,i}, i) = g(e, r_{e,i}, i) - \log(-\log g(\epsilon)) \quad (6)$$

$$\epsilon \sim \text{Uniform}(0,1) \quad (7)$$

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.

$$x_i^{(l)} = e_i^{(l-1)} + \sum_{(e,r,i) \in N_i} g(e, r_{e,i}, i) x_e^{(l-1)} \quad (8)$$

$$x_{i'}^{(l)} = e_i^{(l-1)} + \sum_{(e,r,i) \in N_i} g'(e, r_{e,i}, i) x_e^{(l-1)} \quad (9)$$

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

$$\mathcal{L}_{noise} = -\log \frac{\exp(s(x_i^{(l)}, x_{i'}^{(l)})/\tau)}{\sum_{i=0}^l \exp(s(x_i^{(l)}, x_{i'}^{(l)})/\tau)} \quad (10)$$

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

$$g(e, r_{e,i}, i) = \begin{cases} g(e, r_{e,i}, i) \in \text{top-}k(g(e, r_{e,i}, i)) \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

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:

$$M_k^1 = g(e, r_{e,i}, i) \quad (12)$$

$$M_k^2 = 1 - g(e, r_{e,i}, i) \quad (13)$$

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

$$\eta_1(G_k) = (e, r, i) \odot M_k^1, \eta_2(G_k) = (e, r, i) \odot M_k^2 \quad (14)$$

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

3.4 Local contrastive learning

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

$$c_i = s\left(f_k(x_i, \eta_1(G_k)), f_k(x_i, \eta_2(G_k))\right) \quad (15)$$

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_k^1 and M_k^2 , which are derived from a Bernoulli distribution [24], to perform data augmentation on the user-item interaction view:

$$\varphi(G_u) = (V, M_u^1 \odot Y), \varphi(G_u) = (V, M_u^2 \odot Y) \quad (16)$$

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:

$$e_u^{loc} = e_u^{(0)} + \dots + e_u^{(L)}, e_i^{loc} = e_i^{(0)} + \dots + e_i^{(L)} \quad (17)$$

$$\mathcal{L}_{loc} = \sum_{n \in V} -\log \frac{\exp(s(x_n^1, x_n^2)/\tau)}{\sum_{n' \in V, n' \neq n} \exp(s(x_n^1, x_n^2)/\tau)} \quad (18)$$

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}_{loc} denotes the local contrastive loss function.

3.5 Global contrastive learning

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

$$z_c^{glo} = W^{(2)} \sigma(W^{(1)} e_c^{glo} + b^{(1)}) + b^{(2)} \quad (19)$$

$$z_c^{loc} = W^{(2)} \sigma(W^{(1)} e_c^{loc} + b^{(1)}) + b^{(2)} \quad (20)$$

Where $W \in R^{d \times d}$ and $b \in R^{d \times 1}$ 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:

$$\mathcal{L}_{glo} = -\log \frac{\exp(s(z_c^{glo}, z_c^{loc})/\tau)}{\underbrace{\exp(s(z_c^{glo}, z_c^{loc})/\tau)}_{\text{positive pairs}} + \underbrace{\sum_{k \neq c} \exp(s(z_c^{glo}, z_k^{glo})/\tau)}_{\text{negative pairs within the view}} + \underbrace{\sum_{k \neq c} \exp(s(z_c^{glo}, z_k^{loc})/\tau)}_{\text{negative pairs between the view}}} \quad (21)$$

To combine the recommendation task with the self-supervised task, we adopt a multi-task training strategy

to optimize the entire model. Firstly, the BPR [26] loss \mathcal{L}_{BPR} is constructed, therefore, the primary function after introducing the contrastive loss is as follows:

$$\mathcal{L}_{MVKGCL} = \mathcal{L}_{BPR} + \beta(a(\mathcal{L}_{loc} + \mathcal{L}_{noise}) + (1-a)\mathcal{L}_{glo}) + \lambda \|\theta\|_2^2 \quad (22)$$

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.

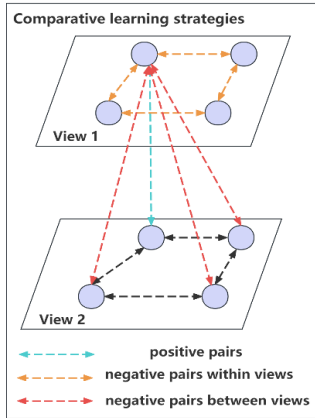


Figure 2: Contrastive learning strategies.

4. Experiments

4.1 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 1 below.

Table 1: Statistics of the datasets used in experiments

	Amazon-book	Yelp2018
User-Item Interactions	#Users	45919
	#Items	45538
	#Interactions	1183610
KG	#Entities	47472
	#Relations	42
	#Triples	869603

4.2 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).

4.3 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.

4.4 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].

4.5 Performance comparison with SOTA

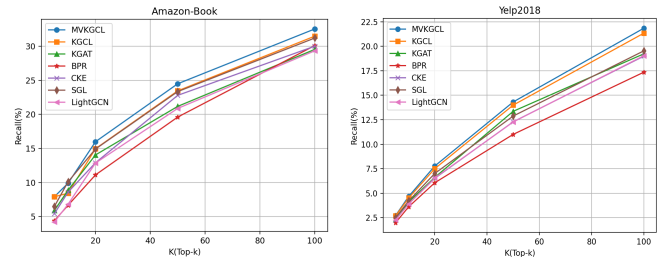


Figure 3: The result of Recall@K in top-K recommendation.

Table 2: Performance results obtained

Model	Amazonb		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

Table 2 above presents the experimental results of all 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.

4.6 Ablation study of MVKGCL

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 3 below.

Table 3: 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

4.7 Impact of local-level contrastive loss weight

Investigating the impact of local versus global contrast weights on model metrics. Specifically, to study the effect of the weight parameter α , the model varies α '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 $\alpha=0.4$; for the Yelp2018 dataset, the optimal performance is reached when $\alpha=0.2$, 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 $\alpha=0$, highlighting the significance of the automatic masking mechanism. For the Yelp2018 dataset, poor performance is observed when the parameter α 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.

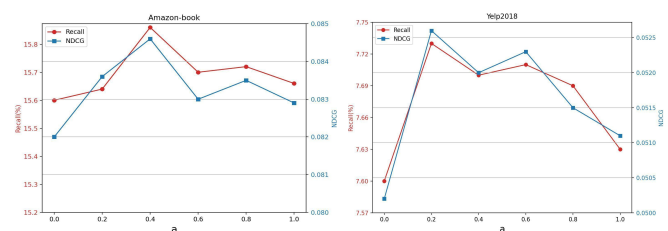


Figure 4: Impact of local-level contrastive loss weight.

4.8 Impact of contrastive loss weight

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 4 below.

Table 4: Impact of contrastive loss

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

4.9 Vector embedding representation

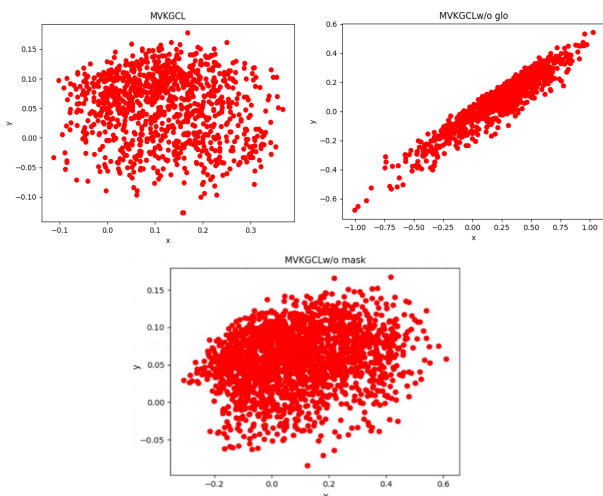


Figure 5: Project Embedding Representation in Amazon-Book.

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.

5. 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.

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 Okana 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 Heterogeneous Graph Neural Network with 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, Danyang 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.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

An Improved DBNet Model with Pyramid Pooling and Multi-Scale Enhancement for Small Text Object Detection

Zhu Yuan^{a,1}, Wuxuan Tang^{b,*}, Jin Yao^{a,3} and Chengjun Liu^{a,4}

^aGuangzhou Metro Design & Research Institute Co., Ltd., Guangzhou, China

^bInstitute of Intelligence Science and Technology, School of Computer Science and Software Engineering

ARTICLE INFO

Submitted 7.31.2024
Revised 8.31.2024
Second Revision 10.31.2024
Accepted 11.15.2024

Keywords:

Small text object detection
DBnet
contextual information fusion
Dual attention

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 dilated 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.

© 2019 KSI Research

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

*Corresponding author

Email address: tangwx2001@hhu.edu.cn

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 significant drawbacks:



Figure 1: An image where 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. Inspired by these

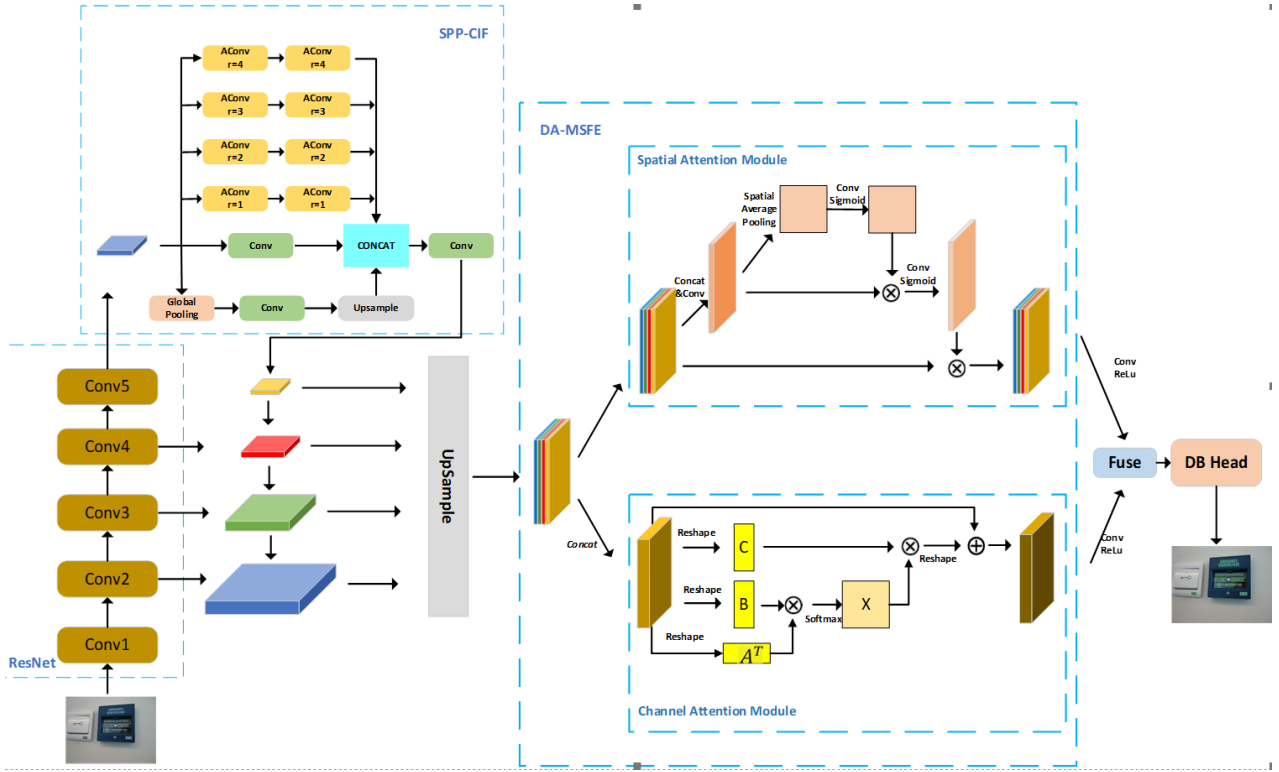


Figure 3: The architecture of the Improved DBNet Model with SPP-CIF and DA-MSFE

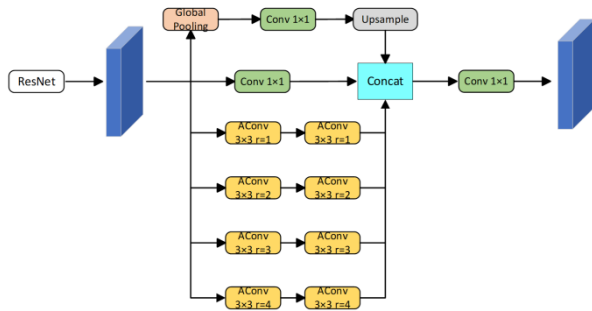


Figure 4: Spatial Pyramid Pooling-based Context Information Fusion Module SPP-CIF.

methods, this section proposes a Pyramid Pooling (SPP-CIF), which can extract information of high-level features from six different receptive fields.

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 Context Information Fusion Module based on Spatial 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 F_{in} information with different receptive fields from three respective paths. The process is as follows:

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

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

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' ($F' \in \mathbb{R}^{\frac{C}{2} \times H \times W}$), preserving some of the original information of the feature map. The formula for this subprocess is as follows:

$$F' = Conv_{1 \times 1}(F_{in}) \quad (2)$$

where $Conv_{1 \times 1}$ 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:

$$A'_i = AConv_{3 \times 3, i} \left(AConv_{3 \times 3, i} (F_{in}) \right), i = 1, 2, 3, 4 \quad (3)$$

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

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 $(F_{out} \in \mathbb{R}^{C \times H \times W})$) integrates both global information extracted from high-level feature maps and contextual information. The subprocess is expressed as follows

$$F_{out} = Conv_{1 \times 1} (Concat(F'_u, F', A'_1, A'_2, A'_3, A'_4)) \quad (4)$$

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.

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 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.

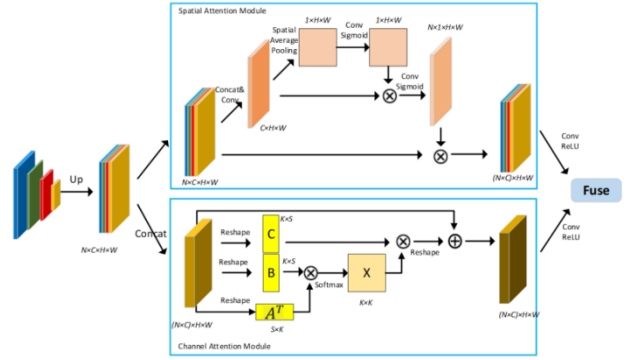


Figure 5: Dual Attention-based Multi-Scale Feature Enhancement.

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:

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

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:

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

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 $Conv_{3 \times 3, ReLU}$ 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 Generalities About the Model

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.

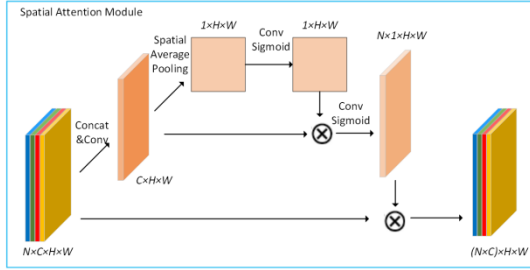


Figure 6: Spatial Attention Module.

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

$$F'_{in} = Conv_{3 \times 3}(Concat(F_{in}^1, F_{in}^2, F_{in}^3, F_{in}^4)) \quad (7)$$

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

(2) Perform global pooling on F'_{in} to obtain F_{avg} ($F_{avg} \in \mathbb{R}^{1 \times H \times W}$), then apply a convolution with kernel size 3×3 to F_{avg} followed by a $Sigmoid$ function to obtain the descriptor M_s ($M_s \in \mathbb{R}^{1 \times H \times W}$). 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:

$$M_s = Sigmoid(Conv_{3 \times 3}(AvgPool(F'_{in}))) \quad (8)$$

(3) Multiply the spatial descriptor M_s with the feature map F_{in} , then apply convolution with kernel size 3×3 to the result followed by a $Sigmoid$ function to obtain the spatial attention A_s ($A_s \in \mathbb{R}^{N \times 1 \times H \times W}$, $N = 4$). This subprocess is expressed as follows:

$$A_s = Sigmoid(Conv_{3 \times 3}(M_s \otimes F'_{in})) \quad (9)$$

(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 $F_{outs} \in \mathbb{R}^{(N \times C) \times H \times W}$. This subprocess is formulated as follows:

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

4.2 Channel Self-Attention for Multi-Scale Feature Enhancement

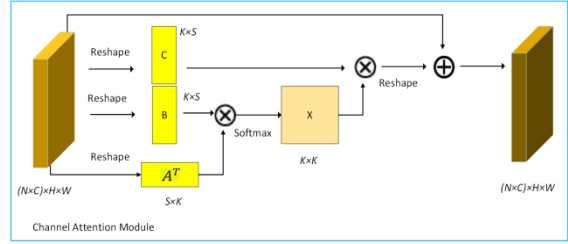


Figure 7: Channel Self-Attention Module.

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

(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 ($X \in \mathbb{R}^{K \times K}$). $X_{i,j}$ 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:

$$X = Softmax(B \times A^T) \quad (11)$$

(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:

$$A_c = X \times C \quad (12)$$

(4) Reshape the feature map A_c into A_c^T ($A_c^T \in \mathbb{R}^{(N \times C) \times H \times W}$), 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:

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

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 from 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:

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

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 Experiments

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	81.3

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	78.6	82.1
2	EAST	PVANet	81.1	72.9	76.8
3	SegLink	VGG-16	72.8	77.00	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.

Figure 8 illustrates some of the detection results of the improved model DBNet-SD and the base model DBNet on the ICDAR2015 dataset. These images feature indoor scenes where text objects are blurred due to light pollution. In the first row of pictures, the text objects under the second icon are detected on the left side, whereas they are missed on the right side. In the second row, more blurred texts are detected on the left side but overlooked on the right side. In the third row, the word “SingTd” in the upper-right corner is found on the left side, while it is not detected on the right side. Therefore, it is evident that the improved model can effectively mitigate the interference from light and other noises, demonstrating better detection capability than the base model.



Figure 8: Examples of detection results on the dataset ICDAR2015. The left and right column corresponds to DBNet-SD and DBNet, respectively.



Figure 9: Examples of detection results on the data set MSRA-TD500. The left and right column corresponds to DBNet-SD and DBNet, respectively.

Figure 9 shows some of the detection results of the improved model DBNet-SD and the base model DBNet on the Figure 8: Examples of detection results on the dataset ICDAR2015. The left and right column corresponds to DBNet-SD and DBNet, respectively. MSRA-TD500 dataset. Long text objects are present in all images. In the first row of images, the left side detects the Chinese texts “ Press Here ” and “Emergency Break” in the images, while the right side does not. In the second row, the word “full” is detected on the left side, while it is not detected on the right side. In the third row, the Chinese word “corporation” is completely detected on the left side, while it is not detected on the right side. Obviously, it suggests that the improved model DBNet-SD can effectively mitigate the issue that the edges of long text objects are frequently undetected in the base model.

From Figure 8 and 9, it can be summarized that both the improved model and the base model exhibit a certain degree of text area miss-detection, but the improved model shows significant effectiveness in reducing miss-detections of small text objects compared to the base model, which implies that the embedded modules SPP-CIF and DA-MSFE can promote the performance of text object detection, especially for the detection of small text objects.

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] Dai, P., Zhang, S., Zhang, H., et al., 2021. Progressive contour regression for arbitrary-shape scene text detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE. pp. 7393-7402.
- [2] Fu, J., Liu, J., Tian, H., et al., 2019. Dual attention network for scene segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE. pp. 3146-3154.
- [3] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 770-778.
- [4] Li, H., Xiong, P., An, J., et al., 2018a. Pyramid attention network for semantic segmentation. arXiv preprint arXiv:1805.10180.
- [5] Li, X., Wang, W., Hou, W., et al., 2018b. Shape robust text detection with progressive scale expansion network. arXiv preprint arXiv:1806.02559.
- [6] Liao, M., Wan, Z., Yao, C., et al., 2020. Real-time scene text detection with differentiable binarization, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI. pp. 11474-11481.
- [7] Liao, M., Zhu, Z., Shi, B., et al., 2018. Rotation-sensitive regression for oriented scene text detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp.5909-5918.
- [8] Liao, M., Zou, Z., Wan, Z., et al., 2022. Real-time scene text detection with differentiable binarization and adaptive scale fusion. IEEE Transactions on Pattern Analysis and Machine Intelligence 45, 919-931.

- [9] Lin, T.Y., Dollár, P., Girshick, R., et al., 2017. Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 2117-2125.
- [10] Long, S., Ruan, J., Zhang, W., et al., 2018. Textsnake: A flexible representation for detecting text of arbitrary shapes, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer. pp. 20-36.
- [11] Shi, B., Bai, X., Belongie, S., 2017. Detecting oriented text in natural images by linking segments, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 2550-2558.
- [12] Tang, J., Zhang, W., Liu, H., et al., 2022. 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, IEEE. pp. 4563-4572.
- [13] Tian, Z., Huang, W., He, T., et al., 2016. Detecting text in natural image with connectionist text proposal network, in: Computer Vision-ECCV 2016: 14th European Conference, Springer. pp. 56-72.
- [14] Woo, S., Park, J., Lee, J.Y., et al., 2018. Cbam: Convolutional block attention module, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer. pp. 3-19.
- [15] Ye, M., Zhang, J., Zhao, S., et al., 2023. Dptext-detr: Towards better scene text detection with dynamic points in transformer, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI. pp.3241-3249.
- [16] Zhang, S., Yang, C., Zhu, X., et al., 2023. Arbitrary shape text detection via boundary transformer. IEEE Transactions on Multimedia.
- [17] Zhang, X., Su, Y., Tripathi, S., et al., 2022. Text spotting transformers, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE. pp. 9519-9528.
- [18] Zhao, H., Shi, J., Qi, X., et al., 2017. Pyramid scene parsing network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 2881-2890.
- [19] Zhou, X., Yao, C., Wen, H., et al., 2017. East: An efficient and accurate scene text detector, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 5551-5560.

Journal of Visual Language and Computing

journal homepage:

Educational Video Ecosystems: An Interactive Mind Map-Driven Approach

Taeghyun Kang^{a,*}, Hyungbae Park^b and Sunae Shin^c

^aUniversity of Central Missouri, Warrensburg, MO, USA

^bUniversity of North Georgia, Dahlonega, GA, USA

^cGeorgia Gwinnett College, Lawrenceville, GA, USA

ARTICLE INFO

Article History:

Submitted August 1, 2024

Revised December 1, 2024

Second Revision December 15, 2024

Accepted December 17, 2024

Keywords:

Video sharing platform

Video searching platform

Mind map

UX design

ABSTRACT

In an era where digital learning is becoming increasingly vital, the need for effective educational tools that can adapt to the demands of diverse learners is becoming more urgent. The speed of the Internet has improved dramatically owing to the growth of Internet-based technologies and networks, which has made video content the most significant medium for distributing information. Additionally, advancements in online video sharing and production platforms such as YouTube, TikTok, and Instagram, have made it easier for individuals to create and share videos. Research shows that video-based learning can increase engagement, critical thinking, and knowledge retention when aligned with learning objectives and interactive elements. 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 present a video search platform built on a mind map architecture. The mind map is an effective and efficient tool and a valuable method for promoting critical thinking. It helps people to start exploring and expanding the topics in a way that mirrors the brain's natural processes. We developed MVP (Mind Map-Based Video Searching Platform), a service that allows users to create mind maps and upload videos to each node within the mind map as well as learners to search for mind maps created by other users or generated by AI.

© 2025 KSI Research

1. Introduction

As 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

*Corresponding author

Email address: tkang@ung.edu

ORCID: 0000-0003-1150-3810

searching for additional information for each node.

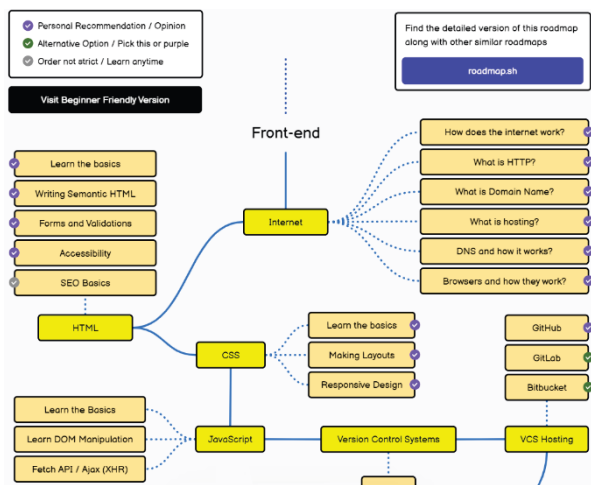


Figure 1: Roadmap of a frontend developer using a mind map structure.

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 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 [23, 11]. 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. [20], Zhang et al. [25], and Zhao et al. [26] 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. [18] proposed a model that learns video-text embeddings from heterogeneous data sources. Similarly, Mithun et al. [19] and Liu et al. [17] proposed a novel framework that utilizes available multi-modal cues from videos for the cross-modal video-text retrieval task. For effective video retrieval, [19] 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. [17] 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, 21, 9, 16]. 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 [22, 15, 27], a user modeling and recommender system [6, 12], document summarization [24, 14], etc. Our proposed platform uses 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.

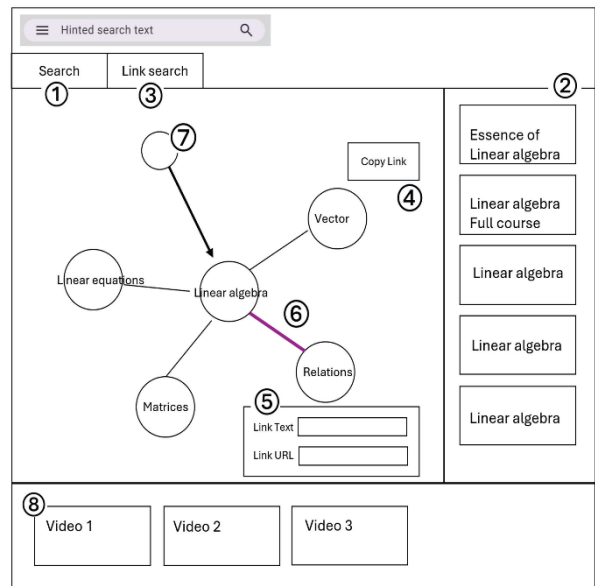


Figure 2: UX design diagram of the mind map.

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 navigate nodes in the mind map to find desired information, the frequency of traversals is represented 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.

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

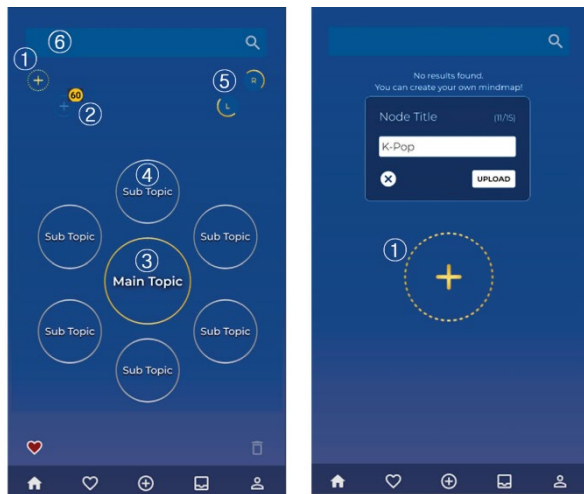


Figure 3: UX design of the mind map in MVP.

Table 3: Functionalities of the mind map for mobile devices

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 created 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.

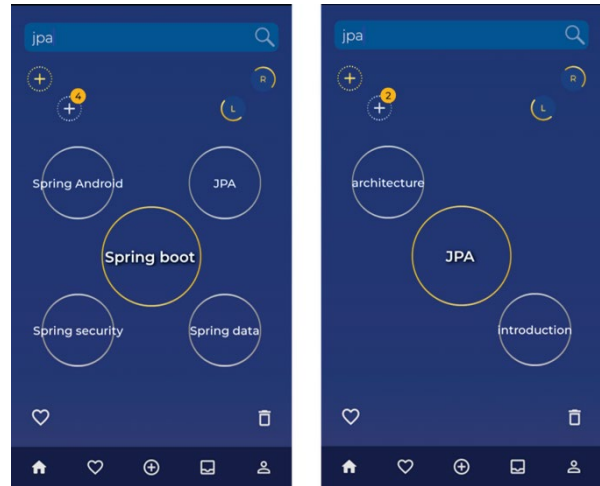


Figure 4: Mind map search outcome of MVP.

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.

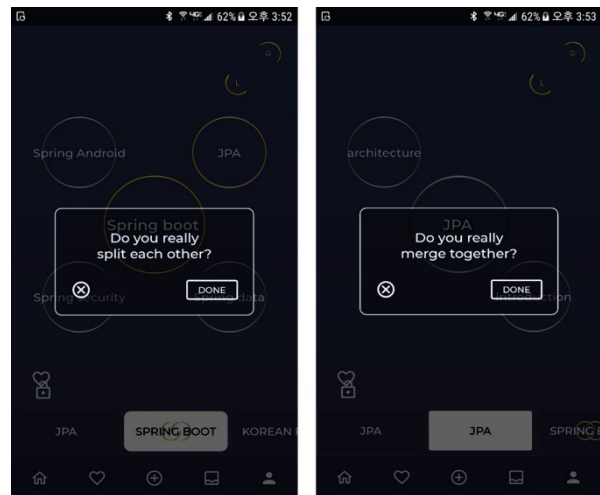


Figure 5: Integration and division of mind map in MVP.

4. Implementation

To collect data to verify the efficiency of MVP as a video sharing platform, the application 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 [13]. 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 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 database
RDS2	MySQL 8.0.15 Community – Read replicas
S3	Secure cloud storage for videos
ElasticCache	Session server – Cache

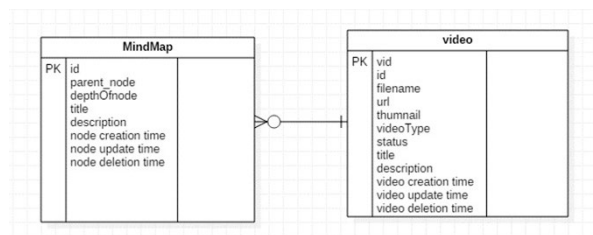


Figure 6: Entity diagram for mind map in MVP.

As illustrated in Figure 7, a desktop-based mind map application was also developed as a proof-of-concept to demonstrate its feasibility and functionality. The interface displays the searched mind map is displayed at the center and related videos are listed at the bottom when a user selects a node on the mind map. Users can customize the mind map by adding additional videos or deleting nodes as needed. For the desktop version, the backend was implemented as a RESTful API using the Spring Boot framework, leveraging the Java Persistence API (JPA) for object-relational mapping and data management, with H2 configured as the database. The frontend was developed with ReactJS, providing a dynamic and user-friendly interface for an enhanced user experience.

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.

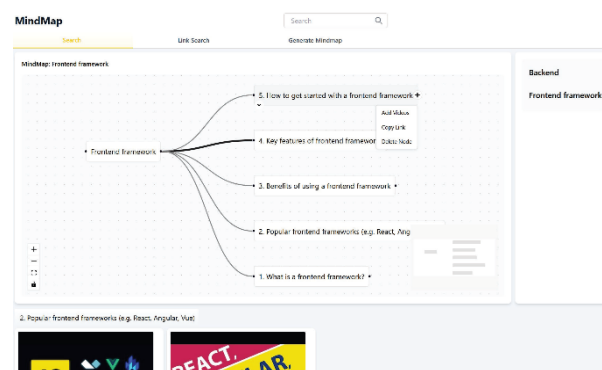


Figure 7: Desktop interface of MVP.

6. Future Work

When learners search for topics, the results will include two types of mind maps: those generated by humans and those created by AI. By comparing the AI-produced results with those created by users in an educational context, learners are encouraged to engage in analytical thinking about the outcomes rather than simply relying on AI-generated results. This process stimulates their creativity and curiosity. We will quantitatively and qualitatively evaluate the impacts of this feature of our platform.

Additionally, to ensure the consistency and integrity of stored mind maps, we will develop conflict resolution mechanisms, data validation processes, and version control features. These measures will maintain data reliability across the system as users modify and expand their mind maps.

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/youtubevideo-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/shareremoved-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] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, et al. The YouTube video recommendation system. In Proceedings of the fourth ACM conference on Recommender systems, pages 293–296, 2010.
- [12] M. H. Dlab. Experiences in using educational recommender system ELARS to support e-learning. In Proceedings of the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pages 672–677. IEEE, 2017.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] A. Miech, I. Laptev, and J. Sivic. Learning a text-video embedding from incomplete and heterogeneous data. arXiv preprint arXiv:1804.02516, 2018.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] D. M. West. The impact of improved internet speeds on information dissemination. https://www.brookings.edu/wp-content/uploads/2016/06/West_Evolution-of-VideoStreaming-and-Digital-Content-Delivery_Final.pdf, 2014.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] S. Zubaidah, N. M. Fuad, S. Mahanal, and E. Suarsini. Improving creative thinking skills of students through differentiated science inquiry integrated with mind map. Journal of Turkish Science Education, 14(4):77–91, 2017.

Journal of Visual Language and Computing

Volume 2024