International Journal of Software and Informatics, ISSN 1673-7288
http://www.ijsi.org, ijsi@iscas.ac.cn, +86-10-62661048
IJSI, 2021, 11(4): 383–403, doi: 10.21655/ijsi.1673-7288.00253

Research Article

Automatic Generation of Large-Granularity Pull Request Description

Li Kuang (邝砾)[1], Ruyi Shi (施如意)[1], Leihao Zhao (赵雷浩)[1], Huan Zhang (张欢)[1], Honghao Gao (高洪皓)[2]

[1] (School of Computer Science and Engineering, Central South University, Changsha 410083, China)
[2] (School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China)
Corresponding author: Honghao Gao, gaohonghao@shu.edu.cn

Abstract In GitHub platform, many project contributors often ignore the descriptions of Pull Requests (PRs) when submitting PRs, making their PRs easily neglected or rejected by reviewers. Therefore, it is necessary to generate PR descriptions automatically to help increase the PR pass rate. The performances of existing PR description generation methods are usually affected by PR granularity, so it is difficult to generate descriptions for large-granularity PRs effectively. For such reasons, this work focuses on generating descriptions for large-granularity PRs. The text information is first preprocessed in PRs and word-sentence heterogeneous graphs are constructed where the words are taken as secondary nodes, so as to establish the connections between PR sentences. Subsequently, feature extraction is performed on the heterogeneous graphs, and then the features are input to a graph neural network for further graph representation learning, from which the sentence nodes can learn more abundant content information through message delivery between nodes. Finally, the sentences with key information are selected to form a PR description. In addition, the supervised learning method cannot be used for training due to the lack of manually labeled tags in the dataset; therefore, reinforcement learning is adopted to guide the generation of PR descriptions. The goal of model training is minimizing the negative expectation of rewards, which does not require the ground truth and directly improves the performance of the results. The experiments are conducted on real dataset and the experimental results show that the proposed method is superior to existing methods in $F1$ and readability.

Keywords Pull Request description; heterogeneous graph neural network; reinforcement learning; unstructured document; summarization generation

Citation Kuang L, Shi RY, Zhao LH, Zhang H, Gao HH. Automatic generation of large-granularity pull request description, *International Journal of Software and Informatics*, 2021, 11(4): 383–403. http://www.ijsi.org/1673-7288/253.htm

GitHub is a leading hosting platform for open-source and private software projects, and more than 5 million users worldwide are hosting more than 10 million open-source projects on it[1]. Pull Request (PR) is the message notification mechanism of this platform, through which project contributors notify project owners of the changes they have made to projects. After PRs

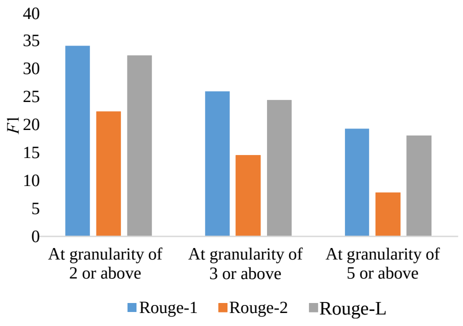
This is the English version of the Chinese article "大粒度 Pull Request 描述自动生成. 软件学报, 2021, 32(6): 1597–1611. doi: 10.13328/j.cnki.jos.006239".

Funding items: National Key R&D Program of China (2018YFB1003800); National Natural Science Foundation of China (61772560)
Received 2020-08-09; Revised 2020-10-26; Accepted 2020-12-19; IJSI published online 2021-12-23

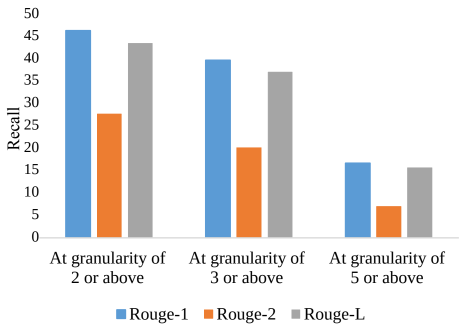
review, project owners can decide whether to merge the changes made by the contributors into the projects according to review results. Research has demonstrated that a good PR description can increase the probability of project changes being merged[2]. However, a large number of PRs on the platform lack descriptions or have over simple descriptions[3], which is not beneficial for the merging of PRs. As such, it is necessary to generate PR descriptions automatically.

Some researchers have modeled the automatic generation task of PR descriptions as a text summarization task. By regarding text information such as commit messages and code comments in PRs as the original documents and PR descriptions as summaries, they used a COder-DECoder (CODEC) model with the attention mechanism to generate PR descriptions and obtained good ROUGE scores as well as manual evaluation scores[3]. However, it was found that the performance of this model was greatly influenced by the PR granularity. A PR may contain one or multiple commits which represent specific changes made by a contributor to a project. In this paper, the number of commits included in a PR is referred to as the granularity of the PR. In the PR dataset used in Reference [3], almost half of PRs contain only two commits. The description generation for PRs with small granularities is very simple, and a short and accurate PR description can be obtained even if only the text information of two commits is listed. As shown in Figure 1, after removing the PRs at granularity of 2 in the dataset, we retrain the model by this method and test it. It is found that the model performance is degraded. When this method is adopted to generate descriptions for PRs at granularity of 5 or above, the performance of this method decreases more in terms of $F1$ and recall. $F1$, recall and precision are evaluation metrics of the text summarization task. $F1$ is the weighted average of precision and recall. Therefore, it is believed that this method is not suitable for generating descriptions for large-granularity PRs, and the method of generating descriptions for large-granularity PRs is waiting to be explored.

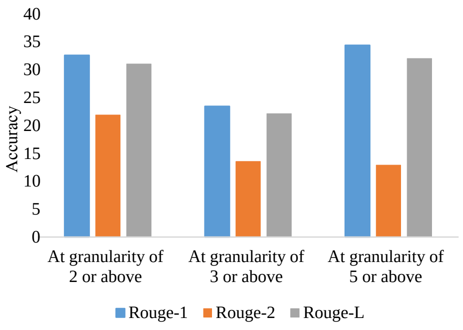
To solve this issue, we model PR description generation as a text summarization problem. The sentences generated by the abstractive text summarization model do not have good readability and are subject to logical errors. Thus, an extractive text summarization model is built in this paper to select the sentences with key information in PR documents to form PR descriptions. However, as indicated by Reference [4], most of the current mainstream extractive text summarization models learned the position information of sentences rather than the content information, so these models performed well on the datasets of structured documents[5], such as news[6]. The importance of sentences in structured documents is generally related closely to sentence positions. For example, in the inverted pyramid structure frequently used in news documents, the sentences in higher positions are more important and more likely included in summaries. In PR documents, however, there is no strong correlation between the importance of a sentence and its position. Therefore, the model should be capable of learning the content of documents and determining the importance of a sentence by its content. Based on the work of Zhong *et al.*[4], Wang *et al.*[7] assumed that graph neural networks were beneficial to helping models learn document content and used them for summarization. They achieved good results, thus verifying this assumption. Inspired by the above work, we model the PR text information as a word-sentence heterogeneous graph and use a graph neural network to further learn the content of sentences in the PR text. Specifically, the word-sentence heterogeneous graph of a PR is taken as input. First, we extract features of nodes and edges in the heterogeneous graph and then input the feature vectors to the graph neural network to further learn the graph structure information and node content information. The feature vector of a sentence node obtained by linear transformation, which is learned by the graph neural network, is taken as the probability of this sentence being included in the PR description. The k sentences with the highest probabilities are selected to form the PR description. In addition, with the REINFORCE algorithm, we avoid using real manually labeled tags to guide the generation of PR descriptions by rewarding the generated PR descriptions and setting the training goal to minimizing negative expectations of rewards. Moreover, we directly optimize the indexes for evaluating the generated PR descriptions, thus improving the model performance.



(a) $F1$



(b) Recall



(c) Accuracy

Figure 1 Performance of the method proposed by Liu *et al.* on the PR data sets at different granularities

In summary, the contribution of this paper consists of the following three aspects.

(1) With regard to the description generation for large-granularity PRs, the PR description generation task is modelled to an extractive text summarization issue and key sentences in PR documents are selected to generate PR descriptions. A PR word-sentence heterogeneous graph is constructed to build the connection between PR sentences, so

as to learn the content of PR documents. Then a graph neural network is employed to transfer messages between nodes, thereby helping the model to explore abundant content features in PR sentences.

(2) Traditional extractive text summarization models require real manually labeled tags provided by datasets for supervised learning, which are not available for PR datasets. Thus, the REINFORCE algorithm is adopted to reward the generated PR descriptions on the basis of evaluation indexes, and the loss function of training is designed as the negative expectation of rewards. As real tags are not related to the design of the loss function, they are not required in training, and the evaluation indexes are directly optimized to make the generated PR descriptions more reliable.

(3) Experiments are conducted on a real dataset. The results demonstrate that the proposed method is effective and performs better than other existing methods.

Section 1 introduces related work. Section 2 discusses the PR description generation method based on graph neural networks. Section 3 describes experimental settings, and Section 4 analyzes experimental results. Section 5 provides the conclusion and an outlook for future work.

1 Related Work

This section introduces related studies on PRs, existing text summarization methods in Natural Language Processing (NLP), and applications of NLP to software engineering.

1.1 Pull Request (PR)

PR is a message notification mechanism on the GitHub platform. When project contributors make changes to a project, they notify the project owner of changes by creating and submitting a PR. The PR reviewer will review the PR submitted by the project contributor. Then, the project owner decides whether to merge the changes into the project on the basis of the review results. Many studies have been developed based on the PR mechanism, such as what PRs are more likely to be merged and new issues arising from the PR mechanism.

The final goal of project contributors creating and submitting PRs is to merge the changes into projects, so many researchers focus on the factors influencing the acceptance of PRs. Soares *et al.*[8] proposed to use the association rule to explore what factors can help changes to be merged into projects. Based on previous studies, Chen *et al.*[9] used a larger dataset to analyze the factors affecting project merging and designed a predictor. The predictor achieved a score of 90% in terms of $F1$. In addition, as the attention on software discrimination has been growing in recent years, some scholars have investigated whether discrimination also exists in merging PRs. For example, Terrell *et al.*[10] investigated the influence of genders on the acceptance of contributions. Jiang *et al.*[11] studied the influence of PR comments on contribution merged and proposed a recommendation method by reviewers.

Some scholars also developed studies on the new issues arising from the PR mechanism. For example, they performed research on estimating the workload of project contributors on the basis of PR information[12], ordering PRs according to their importance[13], and labeling PRs[14]. Moreover, many scholars studied how to recommend PR reviews[15–18]. Liu *et al.*[3] introduced a new direction, the automatic PR description generation. They used a CODEC model integrated with an attention mechanism to solve this issue. However, this method is likely to be affected by the granularity of PRs and fails to generate good descriptions for large-granularity PRs.

1.2 Text summarization

Text summarization is a traditional issue in NLP. According to the generation method, it can be divided into extractive and abstractive summarization. The extractive method directly extracts sentences with key information to form a summary; in the abstractive method, models reorganize sentences according to the learned content of documents to form a summary.

TextRank[19] is a typical extractive summarization method. Referring to the PageRank algorithm[20], this method takes sentences as nodes and constructs undirected and weighted edges according to the similarity between sentences to obtain structure graphs of articles. Then the TextRank scores of the nodes in the graph were calculated, and the sentences represented by N nodes with the highest scores were selected to form the summary. Amid the advancement of neural networks, more and more scholars have started to use neural networks for extractive summarization. Nallapati *et al.*[21] relied on recurrent neural networks to learn the representation vector of a sentence and input it to a classifier, so as to predict whether this sentence should be included in the summary. After that, some scholars also started to explore the applications of other neural networks to extractive summarization. For example, Liu *et al.*[22] adopted Transformer to encode documents. Zhong *et al.*[4] conducted an ablation experiment to investigate the performance of two neural network models, Transformer and LSTM, in an extractive summarization task. The experiment reveals that the two models both mostly learn the position information of sentences rather than the content information, while Transformer can learn more content information than LSTM. On this basis, Wang *et al.*[7] argued that the Transformer-based neural network structure could be regarded as a fully connected graph structure. Therefore, they proposed the conjecture that a graph structure-based model could learn more information about article content. To verify this conjecture, they constructed a structure graph of an article and used graph neural networks to learn the content information of the article, achieving good results. In general, the extractive summarization method is simple and easy to implement. However, the obtained summaries can only be composed of sentences from original texts, and the connections between sentences are rather rigid.

The abstractive method is another method for text summarization. The CODEC model with an attention mechanism[23] is a common model in the abstractive summarization method. Based on this, See *et al.*[24] proposed a pointer generation network to improve this model, which effectively reduced the influence of Out-Of-Vocabulary (OOV) words on the model performance in practical scenarios. Some scholars also improved the original model by changing the design of the attention mechanism. For example, Gehrmann *et al.*[25] designed a bottom-up attention mechanism to enable the model to select more effective information. Liu *et al.*[26] put forward a novel idea for abstractive text summarization. First, a source document was parsed to a set of Abstract Meaning Representation (AMR) graphs. Then the graphs were transformed into a summary graph, based on which the text summary was generated. In addition, Yu *et al.*[27] introduced generative adversarial networks to generate summaries. In conclusion, the summaries generated by the abstractive method have more flexible languages, but they also suffer from the common problems of generating repetitive sentences and poor readability.

As a common method for text summarization, the above method has achieved good results in structured document datasets (such as news datasets). However, unstructured documents do not have evident structures, and the importance of sentences does not correlate well with their positions, so they are quite different from structured documents. Therefore, the above method does not perform well on unstructured document datasets. In this regard, some scholars proposed summarization methods applicable to unstructured document datasets[5, 28–31]. For example, Zhao *et al.*[5] designed a hierarchical adaptive network to address the problem of long documents and the difficulty in capturing effective information in conference document datasets. Liu *et al.*[30] worked on generating summaries for customer service sessions. They believed that the summaries of customer service sessions should be complete and correct in logic, so they proposed to use the Leader-Writer network for generating session summaries. However, the main problems solved by these approaches are highly related to the characteristics of the datasets, so these approaches cannot be directly applied to the description generation for large-granularity PRs.

1.3 Applications of NLP to software engineering

With the successful application of neural networks to translation[32], NLP has developed rapidly and is widely used in many fields including software engineering.

Inspired by the neural network-based translation model[32], many scholars engaged in software engineering have started to think about how to convert specific documents in software engineering, which are highly specialized, into easy-to-read natural language formats for developers[33–36]. For example, some scholars used neural machine translation to automatically generate commit messages, with good results[33, 34]. Some scholars also modelled the generation task of code annotations as a problem of translating the abstract syntax tree of the program source code into code annotations and used a neural network-based translation model to accomplish this task[35, 36].

In addition, other techniques of NLP have also been applied to software engineering. For example, inspired by word embedding, Alon *et al.*[37] proposed the Code2Vec method for embedding code to obtain vector representations of code, which laid a foundation for subsequent research. Ye *et al.*[38] introduced named entity recognition into software engineering and proposed a related method for recognizing specific entities in software engineering. Markovtsev *et al.*[39] proposed a code segmentation method based on the word segmentation algorithm in NLP to further obtain the semantic information in code. Ferrari *et al.*[40] used the word embedding method to detect possible ambiguities in requirements. Chen *et al.*[41] built a Bayesian hierarchical topic model to analyze behavioral trajectories of user-system interactions. Alreshedy *et al.*[42] used NLP techniques to predict the programming language used by questions and code snippets of Stack Overflow. Hao *et al.*[43] employed NLP techniques to process the test reports provided by workers in crowdsourced testing and then aggregated the test reports of crowdsourced testing workers into a comprehensive and complete report for software project managers by clustering.

Automatic generation of PR descriptions has been a new problem proposed by Liu *et al.* in recent years, which can be regarded as a new application of NLP techniques to software engineering. Liu *et al.* found that a large number of PRs on the GitHub platform lacked descriptions, which would be detrimental for PRs to yield good results in reviews[3]. As such, Liu *et al.* modeled the automatic generation of PR descriptions as an abstractive document summarization problem. With the text information in a PR as the source document and a PR description as a summary, they used a CODEC model with an attention mechanism to generate PR descriptions. Based on this, two key problems need to be solved to generate PR descriptions, namely the problem of OOV and the problem of inconsistent training objectives and evaluation indexes. Therefore, scholars introduced a pointer generation network[22] and reinforcement learning to enhance the model performance, and they achieved good results in evaluation indexes and manual reviews. However, as this method did not consider the influence of the PR granularity on results, it failed to generate good descriptions for large-granularity PRs.

2 PR Description Generation Method Based on Graph Neural Networks

Similar to the idea of Liu *et al.*[3], the PR description generation problem is modelled as a summarization problem in this paper, and the text information such as commit messages and code comments in PRs are considered as the source document and PR descriptions as the summary.

However, to avoid logical errors in the generated results, we use an extractive summarization model and generate PR descriptions by sequence labeling. To learn more abundant content information in PR sentences, we construct a word-sentence heterogeneous graph to build the connection between PR sentences. In addition, we use a graph neural network to transmit messages between nodes, so as to further explore content features of PR sentence nodes.

The formal definition of a PR description generation problem is as follows: For a given source document S of a PR, it includes a series of sentences $(s_1, s_2, \cdots, s_n)$. The goal of the PR description generator is to extract k sentences in the document S, so they form a PR description D. For each sentence s_i in the source document S, the generator predicts its tag $y_i \in \{0, 1\}$, where 0 indicates that the sentence is not selected as a description sentence, while 1 indicates the sentence is selected as a description sentence. The overall structure of the PR description generation method based on a graph neural network is shown in Figure 2. We first preprocess the PR description document to build the corresponding word-sentence heterogeneous graph. Then the heterogeneous graph is taken as the input of the PR description generator, and the PR description is generated by the generator. In addition, PR datasets lack real manually labelled tags, making supervised training difficult, so reinforcement learning is adopted to train the model. Specifically, the PR description generator is considered as an agent; the PR source document is considered as a state; the generator selecting sentences to form a PR description is considered as the strategy made by the agent; the generated PR description is considered as a new state. After the PR source document is input to the generator, the generator will make corresponding strategies to generate the PR description. The model uses the reward function to evaluate PR descriptions and adjusts relevant parameters of the generator according to results, so that the model can be trained. This section describes the PR description generation method based on a graph neural network in detail.

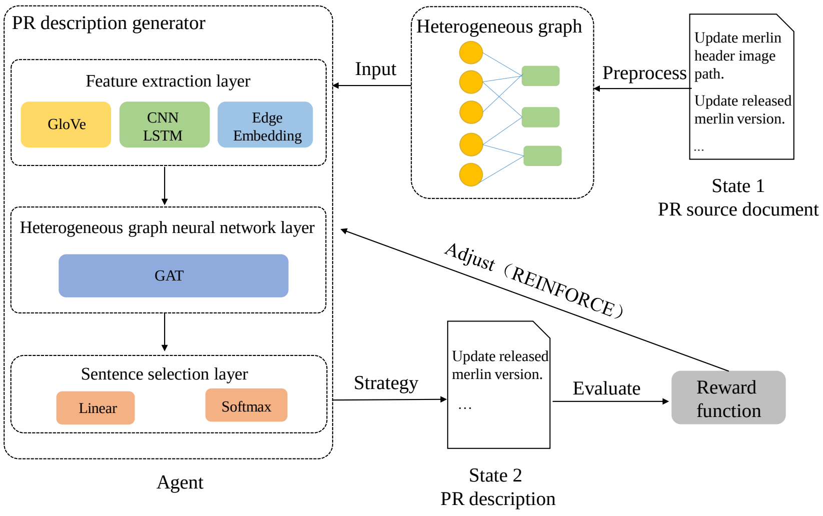


Figure 2 Structure of the PR description generation method

2.1 Graph structure

In this paper, a PR source document is modelled as a weighted heterogeneous graph consisting of word nodes, sentence nodes and their corresponding edges, whose structure is illustrated in Figure 3.

Formally, for a given graph $G = \{V, E\}$, V indicates nodes and E represents edges.

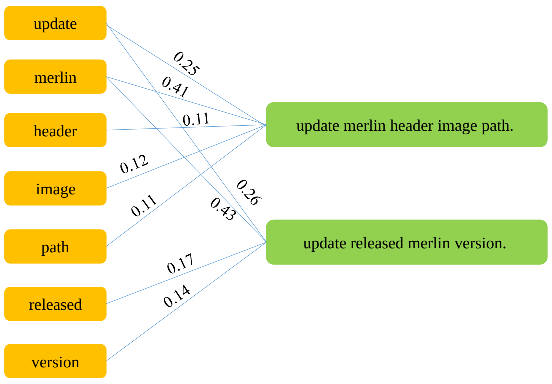


Figure 3 Heterogeneous graph structure

Further, $V = V_s \cup V_w$ is the set of sentence nodes and word nodes. $V_s = \{s_1, s_2, \cdots, s_n\}$ indicates that the sentence node set V_s includes n sentences; V_w is the set of word nodes, and $V_w = \{w_1, w_2, \cdots, w_m\}$ indicates that the set includes m words. In addition, $E = \{e_{11}, \cdots, e_{ij}, \cdots, e_{mn}\}$ (where $i \in \{1, \cdots, m\}, j \in \{1, \cdots, n\}$). When $e_{ij} = 0$, there is no relationship between the word w_i and the sentence s_j; when $e_{ij} \neq 0$, the sentence includes the word. The value of e_{ij} indicates the importance of the word w_i in the sentence s_j, which is calculated as shown in Equations (1) and (2).

$$t_i = TF_i \times IDF_i \tag{1}$$

$$e_{ij} = \frac{c_i t_i}{\sum_{w_z \in s_j} c_z t_z} \tag{2}$$

where t_i is the TF-IDF[44] score of the word w_i in the PR source document S. TF-IDF is a statistical method to assess the importance of a word for a document in a document set or a corpus. TF_i refers to the occurrence number of the word w_i in the PR source document S. IDF_i is the inverse document frequency of the word w_i, which is a measure of the general importance of a word; a larger value indicates this word has a stronger category differentiation ability. The occurrence number c_i of the word w_i in the sentence s_j is multiplied by the TF-IDF score t_i of the word w_i. The product is then divided by the sum of the TF-IDF scores of each word in the sentence s_j multiplied by the occurrence number of this word, so that e_{ij} can be obtained. Here, we do not directly use the TF-IDF score of the word w_i in the sentence s_j as the weight of the edge. This is because most of sentences in PR source documents are relatively short, and using the calculation method of TF-IDF scores in this situation can cause the calculation results to lose the statistical meaning. However, the weights calculated by Equation (2) retain the statistical meaning of TF-IDF scores and reflect the importance of the word w_i in the PR source document to some extent.

2.2 PR description generator based on heterogeneous graph neural networks

In this section, the structure of the PR description generator based on heterogeneous graph neural networks and the function of each part are introduced. The PR description generator mainly includes the feature extraction layer, the content learning layer of the graph neural network, and the sentence selection layer.

First, the features of the heterogeneous graph are extracted in the feature extraction layer. According to the construction method of the heterogeneous graph, the model needs to extract the features of word nodes, sentence nodes and edges in the heterogeneous graph. The word embedding method is adopted to extract the word features. To make the model perform better, we use the pre-trained GloVe[45] vectors as the features of the word nodes. For a sentence node s_j, a Convolutional Neural Network (CNN) is first used to extract its local features, and then the Bidirectional Long Short-Term Memory (BiLSTM) model is employed to obtain its global feature vector. Then the two obtained vectors are put together and taken as the feature vector of this sentence node. For more information, the features of edges are extracted by a randomly initialized embedding layer.

Then the extracted feature vectors of nodes and edges as well as the structural information of the graph are input into the neural network layer, so that the content of the PR document can be learned. In the neural network layer of the graph, the nodes exchange messages to enrich the content of a particular node. A Graph ATtention network (GAT)[46] is built to transmit messages between nodes. h_i indicates the hidden layer state of node i. Figure 4 describes how to get the hidden layer state h_i' of node i at the next layer.

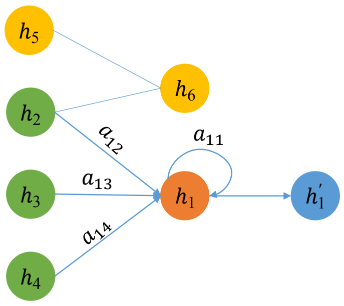


Figure 4 Calculation of hidden layer state h_i' of the graph neural network

With node 1 as an example, the current hidden layer state of node 1 is h_1, and its neighboring nodes are nodes 2–4. Nodes 5 and 6 are in the same subgraph as node 1, but they are not adjacent to node 1. To obtain the new hidden layer state h_1' of node 1, we need to first calculate the attention weights a_{11}–a_{14} of node 1 to itself and to its neighboring nodes 2–4; then, we can acquire the new hidden layer state h_1' through the weighted sum of the hidden layer states of node 1 and its neighboring nodes. At this point, the hidden layer state h_1' transmits the relevant information, containing node 1 and its neighboring nodes, between nodes in this way.

In the neural network layer of the graph, the hidden layer states of sentence nodes are updated by transmitting messages between nodes, so as to obtain the sentence feature vectors with abundant content information. However, the ultimate goal of the model is to predict whether the sentence s_j in the PR source document should appear in the PR description. Therefore, the obtained sentence vectors are processed by a layer of linear transformation to obtain the probability of this sentence in the PR description. The k sentences with the highest probabilities are selected to form the PR description.

2.3 PR description generation method integrated with reinforcement learning

Traditional supervised tag prediction tasks often take the cross-entropy loss function[47] as the loss function of training, which can effectively measure the difference between predicted values and true values. However, in the PR description generation task, using the cross-entropy function as the loss function is not appropriate due to two reasons.

(1) The cross-entropy function is a common loss function in supervised learning, which requires data with real tags. However, PR datasets only include the original text

information of PRs and the real descriptions written by project contributors, without the tag information on whether a sentence in the source document belongs to the PR description. Although the greedy strategy can be adopted to automatically generate tags, as developed in Reference [21], the reliability of this tag generation method is not yet known; if it leads to errors, the risk of error transmission will be introduced to the model.

(2) The ultimate goal is to generate PR descriptions. In line with Liu *et al.*[3], the ROUGE score[48], a common indicator in summarization tasks to measure the difference between generated summaries and real summaries, is taken as an evaluation index to measure the quality of generated PR descriptions. The cross-entropy function, as the loss function, can only measure the prediction accuracy of PR description sentences, but not the fluency of the PR description as a whole, so the overall quality of PR description is not obtained. Therefore, we believe that the cross-entropy function is not suitable as a loss function in this task.

To address the above problems, we introduce reinforcement learning to improve the model. The PR description generator is regarded as an agent, and the PR source document as a state. The PR description generator selecting which sentences to compose a PR description is viewed as the strategy made by the agent, and the generated PR description as a new state. In addition, the reward obtained by the agent is set as the ROUGE score acquired by PR descriptions generated by the agent, and the loss function is defined as the negative expectation of the reward obtained by the agent. Through these settings, the training objective of the model is independent of true values. As a result, it is possible to avoid using true tags to guide the generation of PR descriptions during the training process, and the evaluation indexes are optimized. The above two problems are well solved.

Specifically, the strategy $p(y_i|s_i, S, \theta)$ indicates whether the model selects the sentence $s_i \in S$ as the description sentence, where $y_i \in \{0, 1\}$ and θ is the training parameter. After the agent reads the PR source document S, it will make a decision and obtain the sentence set $\hat{y}$ with $y_i = 1$. The sentences contained in this set are assembled together as the PR description generated by the agent. After that, the agent will obtain a reward $r(\hat{y})$ which can evaluate the results of the PR description generator. The ROUGE score obtained by the description generated by the agent is taken as the reward. The training objective of the model is to minimize the negative expectation of the reward obtained by the agent. Then the loss function $L(\theta)$ can be indicated by Equation (3):

$$L(\theta) = -\mathbb{E}_{\hat{y}\sim p(\theta)}[r(\hat{y})] \tag{3}$$

However, as the reward function, namely the computing mode of the ROUGE score, is not derivable[48], the gradient of the loss function is approximated with the REINFORCE algorithm[49], as shown in Equation (4).

$$\nabla L(\theta) = -\mathbb{E}_{\hat{y}\sim p(\theta)}[r(\hat{y})\nabla \log p(\hat{y}|S, \theta)] \tag{4}$$

where $\nabla L(\theta)$ is the gradient of the loss function $L(\theta)$; $\mathbb{E}$ is the expectation; $r(\hat{y})$ is the reward obtained by the agent for acquiring the set $\hat{y}$; $p(\hat{y}|S, \theta)$ is the probability that the agent obtains the output set $\hat{y}$ with the document S as the input and the parameter of θ. After statistical observations, the REINFORCE algorithm considers that the gradient of the loss function can be approximately replaced by the negative expectation of the product of the gradient of the logarithm of $p(\hat{y}|S, \theta)$ and the reward obtained by $\hat{y}$.

As the strategies made by the agent are diverse, it is very costly to compute all the expectations in the loss function. As such, a single sample is usually used to approximately replace the expectation in practice[32]. Then the gradient of the loss function can be calculated by Equation (5):

$$\nabla L(\theta) \approx -r(\hat{y})\nabla \log p(\hat{y}|S,\theta) \tag{5}$$

where $\hat{y}$ is the single sample that approximately replaces the expectation. According to the REINFORCE algorithm, after the logarithm of the probability $p(\hat{y}|S,\theta)$ of this sample is obtained, the gradient $\nabla L(\theta)$ can be approximately viewed as the negative of the product obtained by multiplying the gradient of this logarithm and the reward $r(\hat{y})$ obtained by this sample.

For a fine model performance and a higher training speed, it is excepted to avoid randomization in the sampling process[50], and the samples with high scores are selected to replace expectations. Therefore, two sampling approaches are used:

(1) The sampling method proposed by Ranzato *et al.*[51] is adopted, which uses a cross-entropy loss function to pre-train a model for obtaining samples with high scores. There are two stages of training in this method. At the first stage, the model is pre-trained by the cross-entropy function. At the second stage, the model parameters are adjusted with the REINFORCE algorithm to obtain the final results.

(2) The sampling method used by Shashi *et al.*[52] is adopted. The search space of $\hat{y}$ in the loss function is restricted to the pre-computed set Y of sentences with high scores. As assumed in other extractive summarization methods, we consider that the selected sentences should have higher ROUGE scores and thus approximate Y as the set of k sentences with the highest ROUGE scores.

3 Experimental Settings

This section describes experimental settings, including the dataset, implementation details, baseline experiments, and evaluation indexes.

3.1 Dataset

The dataset provided by Liu *et al.*[3] is employed, which consists of 333,001 data crawled from 1,000 Java projects on GitHub by Liu *et al.* After filtering, this dataset includes 41,832 valid data. Each data contains a PR source document comprising PR text information such as commit messages and code comments as well as an actual PR description written by project contributors. To analyze the factors that may affect the generation performance of PR descriptions, we counted the number of PRs with different granularities in the dataset. The results are shown in Figure 5.

In Figure 5, the horizontal coordinate represents the granularity of PRs, while the vertical coordinate represents the number of PRs. The distribution of PR granularities is uneven in this dataset, with 17,802 PRs containing only two commits and only 53 large-granularity PRs with a granularity of 20. With the goal of generating descriptions for large-granularity PRs, the PRs with the granularity of 2–4 are filtered and those with the granularity of 5 or above are retained. After filtering, the final dataset contains a total of 10,144 data.

3.2 Implementation details

The vocabulary used in the experiment includes 50,000 words, as that in the conventional settings. To avoid the influences of special words such as stop words on the experimental results, we filtered low-frequency words, stop words and punctuation marks when selecting word nodes.

With the same setting of Wang *et al.*[7], the word nodes are initialized as 300-dimensional feature vectors, sentence nodes as 128-dimensional feature vectors, and the edge e_{ij} as a 50-dimensional vector. In the neural network layer of the graph, the 8-head self-attention mechanism is employed to learn node information and the size of the hidden layer is set to 64. In the training process, the Adam optimizer is used and the learning rate is set to 5e–4. Three sentences are extracted in the decoding process to generate the PR description. The number of extracted sentences is determined by a parameter adjustment experiment, and the experimental results are detailed in Section 4.1.

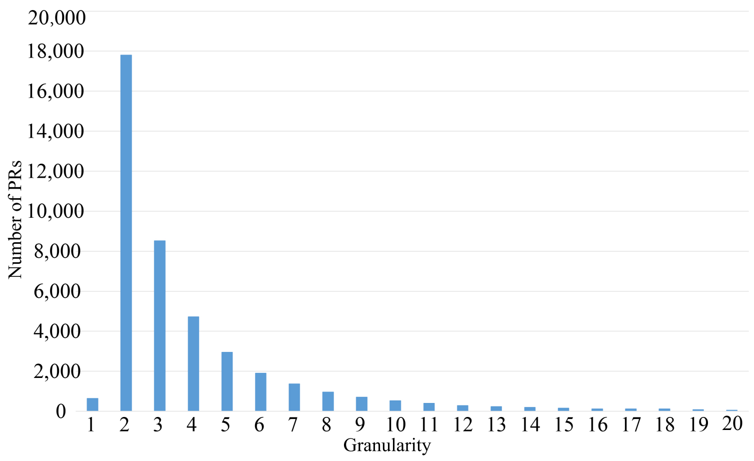


Figure 5 PR granularity statistics

3.3 Baseline experiments

According to Liu *et al.*[3], we selected LeadCM and LexRank as the baseline experiments. Meanwhile, the work of Liu *et al.* was also taken as a baseline experiment to compare with the proposed method.

- LeadCM: It is a common summarization method, which is easily implemented and popular for summarization in the industrial circle. Its idea is to extract the first k sentences of a document to form a summary. With the same settings as the above experiment, this paper also uses LeadCM to obtain three sentences to form a PR description.
- LexRank: It is a summarization method which ranks sentences by calculating their importance and selects the most important sentences in an article as a summary. It represents an application of the PageRank algorithm to extractive document summarization. For a given PR source document, the LexRank method is adopted to rank sentences according to their importance, and then the top three sentences are selected to form a PR description.
- PG+RL: It is the core in the method of Liu *et al.* for generating PR descriptions. This method uses a CODEC model with an attention mechanism to generate PR descriptions abstractly. To solve the problems of massive OOV words and inconsistency between the training objective and evaluation indexes during the generation of PR descriptions, Liu *et al.* introduced pointer generation networks and reinforcement learning to optimize the model.

3.4 Evaluation indexes

In this paper, the ROUGE[44] score, used frequently for summarization, is adopted as an evaluation index to measure the quality of generated PR descriptions. The ROUGE index measures the difference between generated and true summaries in three dimensions: recall, accuracy, and $F1$. It consists of a series of evaluation rules, and the most common ROUGE-N and

ROUGE-L evaluation methods are adopted. The ROUGE-N evaluation method is calculated by

$$R_{\mathit{rouge\text{-}n}} = \frac{\sum_{gen,ref \in S} \sum_{gram_n \in ref} C_{gen}(gram_n)}{\sum_{gen,ref \in S} \sum_{gram_n \in ref} C_{ref}(gram_n)} \tag{6}$$

$$P_{\mathit{rouge\text{-}n}} = \frac{\sum_{gen,ref \in S} \sum_{gram_n \in ref} C_{gen}(gram_n)}{\sum_{gen,ref \in S} \sum_{gram_n \in gen} C_{gen}(gram_n)} \tag{7}$$

$$F1_{\mathit{rouge\text{-}n}} = \frac{2R_{\mathit{rouge\text{-}n}} P_{\mathit{rouge\text{-}n}}}{R_{\mathit{rouge\text{-}n}} + P_{\mathit{rouge\text{-}n}}} \tag{8}$$

where R, P and $F1$ are recall, accuracy and $F1$, respectively; *rouge-*n indicates the ROUGE-N evaluation rule, with n as 1 or 2 and indicating that the n-gram is applied; *gen*, *ref* and S are the generated PR description, the referenced manually-written PR description and the test set, respectively; $gram_n$ is the n-gram phrase; $C_{gen}(gram_n)$ and $C_{ref}(gram_n)$ are the occurrence numbers of n-gram phrases in generated PR descriptions and reference descriptions, respectively.

In this paper, recall measures the proportion of the number of key words (the n-gram phrases in both generated descriptions and reference summaries) in generated descriptions to the reference summary, namely how much information in the reference summary is captured in the generated descriptions. It focuses more on the information included in the generated descriptions. Accuracy measures the proportion of key information in generated descriptions to the generated descriptions. It highlights whether the generated descriptions are concise enough. $F1$ balances recall and accuracy. It is a comprehensive index as it expects generated descriptions to be informative and concise at the same time. In this problem, higher recall is preferred. This means that the model can capture more key information in PRs and the generated results will be more comprehensive in describing PRs. However, focusing only on recall may make the generated results less concise and include massive irrelevant information, so accuracy should also be considered. Thus, the model performance in $F1$ is concerned more after comprehensive consideration.

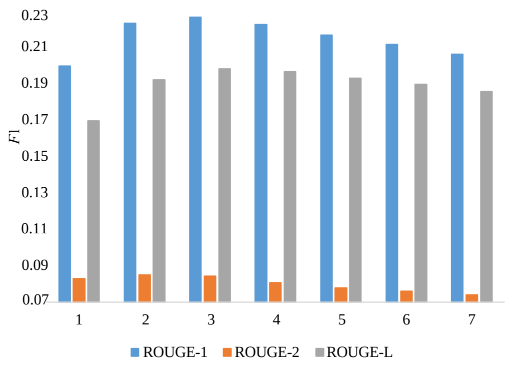
4 Experimental Results

In this section, the experimental results are provided and analyzed. 10-fold cross validation is performed to avoid random results, and the data in this section are the average results of the cross validation.

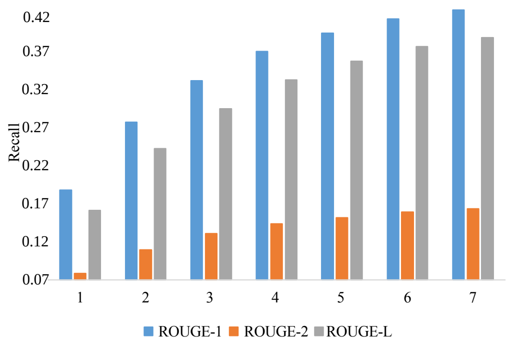
4.1 Parameter adjustment experiment

The parameter adjustment experiment is conducted to verify the most appropriate number of sentences to be extracted in the decoding process. Figure 6 shows the variations of ROUGE scores obtained by the PR descriptions generated by the proposed method as the number of sentences varies. The horizontal coordinate represents the number of extracted sentences, while the vertical coordinate indicates the ROUGE score.

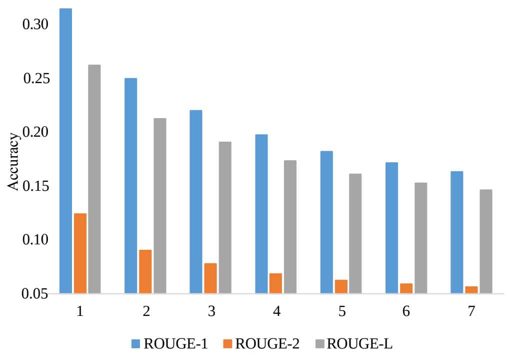
From Figure 6, the recall increases as the number of extracted sentences rises. This is because the probability of getting valid information increases as the number of sentences rises; on the contrary, the accuracy declines, because irrelevant information increases. Therefore, for the balance between the amount of obtained valid information and the conciseness of generated PR descriptions, we focus on the $F1$ dimension of the ROUGE scores for the generated PR descriptions in this section. It is found that in terms of $F1$, the PR descriptions with three extracted sentences have the highest scores. Then three sentences are extracted in the decoding.



(a) $F1$



(b) Recall



(c) Accuracy

Figure 6 ROUGE score changes with the number of extracted sentences

4.2 Comparative experiment

The proposed large-granularity PR generation method based on heterogeneous graph neural network integrated with reinforcement learning is abbreviated as HGPRG-RL. In this section, HGPRG-RL is compared with the baseline methods. The results are shown in Table 1.

The three rows of LeadCM, LaxRank, and PG+RL present the results of the three baseline methods. The row of HGPRS-cross shows the results of the proposed graph neural network based PR description generator with the cross-entropy function as the loss function in the training.

The two rows of HGPRG-RL(A) and HGPRG-RL(B) demonstrate the results of HGPRG-RL with the sampling methods (A) and (B) stated in Section 2.3.

Table 1 Experimental results

Methods	ROUGE-1			ROUGE-2			ROUGE-L		
	$F1$	Recall	Precision	$F1$	Recall	Precision	$F1$	Recall	Precision
LeadCM	22.74	26.68	26.64	**9.6**	12.11	10.14	18.68	22.39	21.19
LexRank	17.85	23.26	19.50	5.23	7.21	5.36	13.89	18.58	14.99
PG+RL	19.34	16.79	**34.29**	7.86	7.05	**12.84**	18.08	15.76	**31.89**
HGPRG-cross	22.59	**32.88**	22.05	8.38	**12.90**	7.77	19.75	**29.21**	19.10
HGPRG-RL(A)	22.74	26.68	26.64	**9.6**	12.11	10.14	18.68	22.39	21.19
HGPRG-RL(B)	**22.83**	32.51	22.61	8.45	12.67	8.44	**19.98**	28.89	19.62

From Table 1, except ROUGE-2, HGPRG-RL(B) can achieve the best results in terms of $F1$, and the proposed method performs well in recall. This means that the proposed method can acquire more useful information than the baseline methods.

It is noted that the method of Liu *et al.* achieves the best results in terms of accuracy. Analysis reveals that it is because this method uses an abstractive summarization model, in which the length of generated PR descriptions is not limited by that of sentences in source documents, and only the keywords in the sentences are required to be extracted. The other methods, including the proposed method, are all extractive ones, in which complete sentences are required to be extracted from the original text. Although the proposed method can capture more key information than the method of Liu *et al.*, the length of generated PR descriptions is limited by original sentences. Therefore, the sentences selected by the model may contain irrelevant information while including substantial valid information, so the accuracy of our method is not as good as that of the method of Liu *et al.* However, our method is still better with regard to the overall ($F1$).

Meanwhile, the accuracy can be improved if reinforcement learning is adopted at the expense of recall, compared with that with the cross-entropy function as the loss function. However, this expense is worthwhile, because the model is optimized by reinforcement learning regarding the overall ($F1$). Moreover, the experiment shows that the sampling method (A) degrades our method to LeadCM. We infer that this is because the model with the cross-entropy function as the loss function has a higher probability of selecting the first three sentences of source documents. This tendency is reinforced after the model parameters are further adjusted by reinforcement learning, so the model is degraded into LeadCM.

4.3 Case analysis

In this section, case study is performed to verify the superiority of the proposed method in the description generation for large-granularity PRs.

(1) Case 1

Original text of PR:

```
Added value ‘unknown’ for ‘repository depth option’. ⟨cm-sep⟩ added a test
case for verifying that depth ‘unknown’ works. ⟨cm-sep⟩ [jenkins-0] changed
‘undefined’ to ‘as-it-is’ in Web interface of subversion-plugin. ⟨cm-sep⟩
[jenkins-0] move ‘as-it-is’ option to the end. ⟨cm-sep⟩ [jenkins-0] updated help
page so it refers to ‘as-it-is’ instead uf ‘unknown’. ⟨para-sep⟩ enable version
mode. Do initial update with infinite depth and check that subdir exists.
Simulate job using ‘svn update-set-depth = files’ and check that subdir no longer
exists. Trigger new build with depth unknown and check that subdir still does
not exist.
```

True PR description:

`I added value ‘unknown’ for ‘repository depth option’ in subversion-plugin this allows a job to reduce size of working copy by executing ‘svn update-set-depth = ··· ’ and have this reduction preserved when job runs again on the same node.`

HGPRG-RL:

`Added value ‘unknown’ for ‘repository depth option’.`

`Enable version mode do initial update with infinite depth and check that subdir exists simulate job using ‘svnupdate-set-depth = files’ and check that subdir no longer exists trigger new build with depth unknown and check that subdir still does not exist.`

`[jenkins-0] updated help page so it refers to ‘as-it-is’ instead uf ‘unknown’.`

PG+RL:

`Added value ‘unknown unknown’ for ‘repository depth option’.`

In Case 1, this PR source document includes five commit messages and one code comment. The true PR description is manually written by the project contributor creating this PR. From the true PR description, the main change made in this PR is the addition of the “unknown” value to “`repository depth option`” in “`subversion-plugin`”. It aims to allow tasks to reduce the amount of content to be copied by executing “`svn update-set-depth = ···`”. In addition, if tasks are executed again at the same node, the amount of copied content is also reduced. In this way, memory consumption can be decreased and the model runs more efficiently. PG+RL is the PR description generated by the method of Liu *et al.* This method only captures the valid information that the “`unknown`” value is added to “`repository depth option`”. Moreover, it repeatedly generates the word “`unknown`”. HGPRG-RL is the PR description generated by the proposed method. It captures the key information of adding the “`unknown`” value to “`repository depth option`” and checking whether the “`subdir exists.simulate`” job uses “`svn update-set-depth = files`”.

(2) Case 2

The original text of PR:

`At a stateless pure javascript/client side component. ⟨cm-sep⟩ wrap models to be able to user resource models. ⟨cm-sep⟩ add an example fo client side tabs. ⟨cm-sep⟩ get rid of extra iteration of tabs. ⟨cm-sep⟩ introduce isactive. ⟨cm-sep⟩ rename model. ⟨cm-sep⟩ make tabs dynamic. ⟨cm-sep⟩ get rid of un-needed space. ⟨cm-sep⟩ improve text. ⟨cm-sep⟩ improve java doc, rename some methods, index start counting at 0 to be compatible with classical tabeedpanel. ⟨cm-sep⟩ add a test unit for client side tabs. ⟨cm-sep⟩ add more tests and add css class specific to each tab. ⟨para-sep⟩ a pure client side tab. it is completely stateless. ⟨para-sep⟩ ⟨p⟩ a ‘pure’ client side stateless tabs component. But instead of generating links that trigger server round trips id does generates just ‘client side’ links. ⟨/p⟩.`

True PR description:

`At a stateless pure javascript/client side component.`

`I just need this for a project and thought it might be useful to other users. It works as in classical but user only needs to use it as ‘standard’ wicket tabs. Mind that all content panels are created at the same time. If accepted i can add an example to demo application.`

HGPRG-RL:

`At a stateless pure javascript/client side component.`

`Wrap models to be able to user resource models.`

`Improve java doc rename some methods index start counting at 0 to be compatible with classical tabeedpanel.`

PG+RL:

`Add an example to be able to user to be compatible with classical tabeedpanel.`

Case 2 is a large-granularity PR consisting of 12 commit messages and 2 code comments. As indicated by the true PR description written by the project contributor, the change made in this PR is to add a user-specific operation to the stateless pure javascript/client-side component. This operation works in the same way as a traditional operation, and users can regard it as a standard tab. The PR description generated with the proposed method obtains the position of this modification. In addition, it also learns that this function is modified for the user resource model and captures some detailed modifications made in this PR.

The PR description generated by the method of Liu *et al.* is analyzed in detail. It reveals that this description originates from the three commit messages "`add an example fo client side tabs`", "`wrap models to be able to user resource models`" and "`improve java doc, rename some methods, index start counting at 0 to be compatible with classical tabeedpanel`". The key words are extracted from three commit messages and simply jointed together. However, the meaning of the jointed sentence is completely different from that of the original one. The purpose of adding the case by the project contributor is not to make users compatible with the original panel. Moreover, it is not linguistically logical for users to be compatible with the panel.

In summary, the PR descriptions generated by the proposed method have high readability. It can avoid some logical errors and capture more useful information. As such, the proposed method performs better in the description generation for large-granularity PRs.

5 Conclusion

To overcome the shortcomings of previous work, a practical method of generating descriptions is proposed for large-granularity PRs. The description generation for large-granularity PRs is modelled to an extractive summarization issue. For the better learning of the content in PR source documents, a word-sentence heterogeneous graph is constructed with word nodes as auxiliary nodes, so that the relationships between sentences in PR source documents can be established. Subsequently, the node feature information of the heterogeneous graph is extracted, and a graph neural network is adopted to learn the graph representation vectors of the PR heterogeneous graph. As a result, the model learns abundant information about PR sentences. Meanwhile, the REINFORCE algorithm is employed to avoid using manually labelled tags to guide the generation of PR descriptions, thus reducing the requirements on datasets. In addition, this enables a better model performance in evaluation indexes. The experimental results on a real dataset show that the proposed method performs better than the existing methods for generating PR descriptions. In the future, we will investigate how to reduce the impact of unregistered words on the content learning of PR source documents to generate better PR descriptions.

References

[1] https://github.com

[2] Georgios G, Storey MA, Bacchelli A. Work practices and challenges in pull-based development: the contributor's perspective. In: Kellenberger P, ed. Proc. of the 38th Int'l Conf. on Software Engineering (ICSE). Austin: IEEE, 2016. 285–296. [doi: 10.1145/2884781.2884826]

[3] Liu ZX, Xia X, Treude C, *et al*. Automatic generation of pull request descriptions. Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). San Diego: IEEE, 2019. 176–188.[doi: 10.1109/ASE.2019.00026]

[4] Zhong M, Liu PF, Wang DQ, *et al*. Searching for effective neural extractive summarization: What works and what's next. Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: Association for Computational Linguistics, 2019. 1049–1058. [doi: 10.18653/v1/P19-1100]

[5] Zhou Z, Pan HJ, Fan CJ, *et al*. Abstractive meeting summarization via hierarchical adaptive segmental network learning. Proc. of the World Wide Web Conf. (WWW 2019). New York: Association for Computing Machinery, 2019. 455–3461. [doi: 10.1145/3308558.3313619]

[6] Kedzie C, Kathleen M, Hal D. Content selection in deep learning models of summarization. Proc. of the Conf. on Empirical Methods in Natural Language Processing. Brussels: Association for Computational Linguistics, 2018. 1818–1828. [doi: 10.18653/v1/D18-1208]

[7] Wang DQ, Liu PF, Zheng YY, *et al*. Heterogeneous graph neural networks for extractive document summarization. Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020. 6209–6219. [doi: 10.18653/v1/2020.acl-main.553]

[8] Soares DM, de Lima Júnior ML, Murta L, *et al*. Acceptance factors of pull requests in open-source projects. Proc. of the 30th Annual ACM Symp. on Applied Computing. New York: Association for Computing Machinery, 2015. 1541–1546. [doi: 10.1145/2695664.2695856]

[9] Chen D, Stolee KT, Menzies T. Replication can improve prior results: A Github study of pull request acceptance. Proc. of the 27th Int'l Conf. on Program Comprehension (ICPC). Montreal: IEEE 2019. 179–190.

[10] Terrell J, Kofink A, Middleton J, *et al*. Gender differences and bias in open source: Pull request acceptance of women versus men. PeerJ Computer Science, 2017, 3: e111. [doi: 10.7717/peerj-cs.111]

[11] Jiang J, Yang Y, He J, *et al*. Who should comment on this pull request? Analyzing attributes for more accurate commenter recommendation in pull-based development. Information and Software Technology, 2017, 84: 48–62.

[12] Maddila C, Bansal C, Nagappan N. Predicting pull request completion time: A case study on large scale cloud services. Proc. of the 27th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. New York: Association for Computing Machinery, 2019. 874–882.

[13] van der Veen E, Gousios G, Zaidman A. Automatically prioritizing pull requests. Proc. of the 12th Working Conf. on Mining Software Repositories. Florence: IEEE, 2015. 357–361. [doi: 10.1109/MSR.2015.40]

[14] Yu S, Xu L, Zhang Y, *et al*. NBSL: A supervised classification model of pull request in Github. Proc. of the IEEE Int'l Conf. on Communications (ICC). Kansas City: IEEE, 2018. 1–6. [doi: 10.1109/ICC.2018.8422103]

[15] Xia X, Lo D, Wang X, *et al*. Who should review this change? Putting text and file location analyses together for more accurate recommendations. Proc. of the Int'l Conf. on Software Maintenance and Evolution (ICSME). Bremen: IEEE, 2015. 261–270. [doi: 10.1109/ICSM.2015.7332472]

[16] Zanjani MB, Kagdi H, Bird C. Automatically recommending peer reviewers in modern code review. IEEE Trans. on Software Engineering, 2016, 42(6): 530–543. [doi: 10.1109/TSE.2015.2500238]

[17] Lu S, Yang D, Hu J, *et al*. Code reviewer recommendation based on time and impact factor for pull request in Github. Ji Suan Ji Xi Tong Ying Yong/Computer Systems Applications, 2016, 25(12): 155–161.

[18] Liao ZF, Wu ZX, Wu JS, *et al*. TIRR: A code reviewer recommendation algorithm with topic model and reviewer influence. Proc. of the 2019 IEEE Global Communications Conf. (GLOBECOM). Waikoloa: IEEE, 2019. 1–6.

[19] Mihalcea R, Tarau P. Textrank: Bringing order into text. Proc. of the 2004 Conf. on Empirical Methods in Natural Language Processing. Barcelona: Association for Computational Linguistics, 2004. 404–411.

[20] Page L, Brin S, Motwani R, *et al*. The PageRank Citation Ranking: Bringing Order to the Web. Stanford InfoLab, 1999.

[21] Nallapati R, Zhai FF, Zhou BW. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. Proc. of the 31st AAAI Conf. on Artificial Intelligence (AAAI 2017). San Francisco: AAAI, 2017. 3075–3081.

[22] Liu Y, Lapata M. Text summarization with pretrained encoders. Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: Association for Computational Linguistics, 2019. 3730–3740. [doi: 10.18653/v1/D19-1387]

[23] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. Proc. of the 3rd Int'l Conf. on Learning Representations (ICLR 2015). 2015.

[24] See A, Liu PJ, Manning CD. Get to the point: Summarization with pointer-generator networks. Proc. of the Annual Meeting of the Association for Computational Linguistics (Vol.1: Long Papers). Vancouver: Association for Computational Linguistics, 2017. 1073–1083. [doi: 10.18653/v1/P17-1099]

[25] Gehrmann S, Deng YT, Rush AM. Bottom-Up abstractive summarization. Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing. Brussels: Association for Computational Linguistics, 2018. 4098–4109.

[26] Liu F, Flanigan J, Thomson S, *et al*. Toward abstractive summarization using semantic representations. Proc. of the 2015 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Denver: Association for Computational Linguistics, 2015. 1077–1086.

[27] Yu LT, Zhang WN, Yu Y. Seqgan: Sequence generative adversarial nets with policy gradient. Proc. of the 31st AAAI Conf. on Artificial Intelligence (AAAI 2017). San Francisco: AAAI, 2017. 2852–2858.

[28] Fumio N, Nakano YI, Takase Y. Predicting meeting extracts in group discussions using multimodal convolutional neural networks. Proc. of the 19th ACM Int'l Conf. on Multimodal Interaction. New York: Association for Computing Machinery, 2017. 421–425. [doi: 10.1145/3136755.3136803]

[29] Pan H, Zhou JP, Zhao Z, *et al*. Dial2desc: End-to-end dialogue description generation. arXiv: 1811. 00185, 2018.

[30] Liu CY, Wang P, Xu J, *et al*. Automatic dialogue summary generation for customer service. Proc. of the 25th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. New York: Association for Computing Machinery, 2019. 1957–1965. [doi: 10.1145/3292500.3330683]

[31] Tao X, Zhang XX, Guo SL, *et al*. Automatic summarization of user-generated content in academic Q&A community based on Word2Vec and MMR. Shu Ju Fen Xi Yu Zhi Shi Fa Xian/Data Analysis and Knowledge Discovery, 2020, 4(4): 109–118.

[32] Kyunghyun C, Merriënboer BV, Gulcehre C, *et al*. Learning phrase representations using RNN encoder-decoder for statistical machine translation. Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Doha: Association for Computational Linguistics, 2014. 1724–1734.

[33] Jiang SY, Armaly A, McMillan C. Automatically generating commit messages from diffs using neural machine translation. Proc. of the 2017 32nd IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Urbana: IEEE, 2017. 135–146.

[34] Xu SB, Yao Y, Xu F, *et al*. Commit message generation for source code changes. Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence (IJCAI). 2019. 3975–3981.

[35] Hu X, Li G, Xia X, *et al*. Deep code comment generation. Proc. of the 2018 IEEE/ACM 26th Int'l Conf. on Program Comprehension (ICPC). Gothenburg: IEEE, 2018. 200–210.

[36] Hu X, Li G, Xia X, *et al*. Deep code comment generation with hybrid lexical and syntactical information. Empirical Software Engineering, 2020, 25: 2179–2217. [doi: 10.1007/s10664-019-09730-9]

[37] Alon U, Zilberstein M, Levy O, *et al*. code2vec: Learning distributed representations of code. Proc. of the ACM on Programming Languages, 2019, 3: 1–29. [doi: 10.1145/3290353]

[38] Ye DH, Xing ZC, Foo CY, *et al*. Software-Specific named entity recognition in software engineering

social content. Proc. of the 2016 IEEE 23rd Int'l Conf. on Software Analysis, Evolution, and Reengineering (SANER). Suita: IEEE, 2016. 90–101. [doi: 10.1109/SANER.2016.10]

[39] Markovtsev V, Long W, Bulychev E, *et al*. Splitting source code identifiers using bidirectional LSTM recurrent neural network. arXiv: 1805.11651, 2018.

[40] Ferrari A., Esuli A. An NLP approach for cross-domain ambiguity detection in requirements engineering. Automated Software Engineering, 2019, 26: 559–598. [doi: 10.1007/s10515-019-00261-7]

[41] Chen H, Damevski K, Shepherd D, *et al*. Modeling hierarchical usage context for software exceptions based on interaction data. Automated Software Engineering, 2019, 26: 733–756. [doi: 10.1007/s10515-019-00265-3]

[42] Alreshedy K, Dharmaretnam D, German DM, *et al*. SCC++: Predicting the programming language of questions and snippets of StackOverflow. Journal of Systems and Software, 2020, 162: 110505. [doi: 10.1016/j.jss.2019.110505]

[43] Hao R, Feng Y, Jones JA, *et al*. CTRAS: Crowdsourced test report aggregation and summarization. Proc. of the 2019 IEEE/ACM 41st Int'l Conf. on Software Engineering (ICSE). Montreal: IEEE, 2019. 900–911.

[44] Shi CY, Xu CJ, Yang XJ. Study of TFIDF algorithm. Ji Suan Ji Ying Yong/Journal of Computer Applications, 2009, 29(z1): 167–170, 180.

[45] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Doha: Association for Computational Linguistics, 2014. 1532–1543. [doi: 10.3115/v1/D14-1162]

[46] Veličković P, Cucurull G, Casanova A, *et al*. Graph attention networks. arXiv: 1710.10903, 2017.

[47] de Boer PT, Kroese DP, Mannor S, *et al*. A tutorial on the cross-entropy method. Annals of Operations Research, 2005, 134(1): 19–67.

[48] Lin CY. Rouge: A package for automatic evaluation of summaries. Proc. of the Text Summarization Branches Out. Barcelona: Association for Computational Linguistics, 2004. 74–81.

[49] Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn, 1992, 8: 229–256. [doi: 10.1007/BF00992696]

[50] Zhang XX, Lapata M, Wei FR, *et al*. Neural latent extractive document summarization. Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing. Brussels: Association for Computational Linguistics, 2018. 779–784. [doi: 10.18653/v1/D18-1088]

[51] Ranzato MA, Chopra S, Auli M, *et al*. Sequence level training with recurrent neural networks. arXiv: 1511. 06732 [cs.LG], 2015.

[52] Narayan S, Cohen SB, Lapata M. Ranking sentences for extractive summarization with reinforcement learning. Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol.1 (Long Papers). New Orleans: Association for Computational Linguistics, 2018. 1747–1759.

Li Kuang, Ph.D., professor, Ph.D. supervisor, CCF member. Her research interests include service computing, swarm intelligence software ecosystems, and machine learning.

Ruyi Shi, master. Her research interests include service computing and swarm intelligence software ecosystems.

Leihao Zhao, bachelor. His research interest is service computing.

Honghao Gao, Ph.D., associate professor, Ph.D. supervisor, CCF senior member. His research interests include formal verification of software, collaborative service computing, wireless networks, Industrial Internet of Things, and intelligent medical image processing.

Huan Zhang, Ph.D. Her research interests include machine learning, data mining, and swarm intelligence software ecosystems.