The Journal of Systems & Software 185 (2022) 111160

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

PRHAN: Automated Pull Request Description Generation Based on Hybrid Attention Network☆

Sen Fang [a], Tao Zhang [a,*], You-Shuai Tan [a], Zhou Xu [b], Zhi-Xin Yuan [c], Ling-Ze Meng [a]

[a] Faculty of Information Technology, Macau University of Science and Technology, Macau, China
[b] School of Big Data and Software Engineering, Chongqing University, Chongqing, China
[c] School of Computer Science and Information Engineering, Hubei University, Wuhan, Hubei, China

ARTICLE INFO

Article history:
Received 28 June 2021
Received in revised form 10 October 2021
Accepted 23 November 2021
Available online 7 December 2021

Keywords:
PR description
Hybrid attention
Byte-pair encoding
Label smoothing

ABSTRACT

Descriptions of pull requests (**PRs**) are posted by developers for describing the modifications that they have made and the corresponding reasons in these PRs. Although PRs help developers improve the development efficiency, some developers usually ignore writing the descriptions for PRs. To alleviate the above problem, researchers generally utilize text summarization model to automatically generate descriptions for PRs. However, current RNN-based models still face the challenges such as low efficiency and out-of-vocabulary (OOV), which may influence the further performance improvement to their models. To break this bottleneck, we propose a novel model aiming at the above challenges, named PRHAN (**P**ull **R**equests Description Generation Based on **H**ybrid **A**ttention **N**etwork). Specifically, the core of PRHAN is the hybrid attention network, which has faster execution efficiency than RNN-based model. Moreover, we address the OOV problem by the utilizing byte-pair encoding algorithm to build a vocabulary at the sub-word level. Such a vocabulary can represent the OOV words by combining sub-word units. To reduce the sensitivity of the model, we take a simple but effective method into the cross-entropy loss function, named label smoothing. We choose three baseline models, including LeadCM, Transformer and the state-of-the-art model built by *Liu et al.* and evaluate all the models on the open-source dataset through ROUGE, BLEU, and human evaluation. The experimental results demonstrate that PRHAN is more effective than baselines. Moreover, PRHAN can execute faster than the state-of-the-art model proposed by *Liu et al.*

1. Introduction

Pull requests (from hereon, PR or PRs) can let developers show others changes that they have pushed to a branch in a repository on modern collaborative coding platforms, e.g., GitHub (Gousios et al., 2014, 2016, 2015). Once a PR is opened, developers can discuss and review the potential changes with collaborators and add follow-up commits before the changes are merged into the base branch.[1] Therefore, in the collaborative coding platforms with Git-based technique, developers can make contributions to a project by pushing a pull request although she/he has no access to the central repository. Then this pull request will be verified and reviewed by core collaborators before being merged into the central repository (Yu et al., 2015).

Generally, a PR is composed of its corresponding description and one or more related commits. As shown in Fig. 1, this PR is collected in the NumPy project.[2] From the content in this PR, we can know what changes are made by developers, i.e., "add examples section", and its corresponding motivation, i.e., "for rfft2 and irfft2 docstring". Furthermore, this PR also contains two commits, in which Commit 1 has no code comment and Commit 2 has the corresponding code comment. A complete PR enables reviewers and developers to quickly understand the changes made for the project, which can improve the development efficiency of projects. However, according to the investigation shown in the literature (Liu et al., 2019b), more than one-third of PRs are incomplete because some developers usually omit to write the descriptions for PRs, which may decrease the development efficiency. Therefore, how to construct an effective model to automatically generate PR descriptions becomes an important problem.

To solve the above issue, Liu et al. (2019b) transformed the PR description generation into the task of text summarization.

☆ Editor: Shane McIntosh.

* Corresponding author.
E-mail addresses: 2009853WII30002@student.must.edu.mo (S. Fang), tazhang@must.edu.mo (T. Zhang), 2109853jim20001@student.must.edu.mo (Y.-S. Tan), zhouxullx@cqu.edu.cn (Z. Xu), yzx@stu.hubu.edu.cn (Z.-X. Yuan), 1909853dii20001@student.must.edu.mo (L.-Z. Meng).

[1] Github Documents for Pull Request.

[2] NumPy.

https://doi.org/10.1016/j.jss.2021.111160
0164-1212/

Example

Description:
DOC: added examples section for rfft2 and irfft2 docstring

Commit 1:
Commit Message: DOC: added examples for rfft2 and irfft2 docstring
Added Code Comments: N/A
Commit 2:
Commit Message: DOC: update rfft2/irfft2 examples
Added Code Comments: Update rfft2 example to conform to fft2 example and switch the irfft2 example to illustrate round-tripping.

Fig. 1. An example of PR in the NumPy project.

Similar to technologies used in text summarization, they integrated attention mechanism (Bahdanau et al., 2016; Luong et al., 2015), pointer network (Gu et al., 2016; See et al., 2017; Vinyals et al., 2015), and self-critical sequence training (SCST) (Rennie et al., 2017), then constructed the first seq2seq model (Sutskever et al., 2014) that generates the PR description by summarizing its commits. In their collected dataset, they achieved competitive experimental results at the metric of ROUGE (Lin, 2004). However, we find that their approach performs badly on the BLEU metric. Intuitively, an effective model should perform well on all relevant metrics. For example, Fernandes et al. (2021) used ROUGE and BLEU metrics to evaluate the effectiveness of their proposed model on the code summarization task. Undoubtedly, their model achieves competitive results both on ROUGE and BLEU metrics. According to our investigation, in the testing dataset, there are about 70% inference PR summarizations that are longer (at least 1.5 times) than that generated by *Liu et al.*'s approach. In contrast to the ROUGE metric, the BLEU metric introduces the brevity penalty to penalize generated texts that are short than their reference texts (Papineni et al., 2002). Therefore, only using the ROUGE metric cannot comprehensively reflect the effectiveness of the model and there is still room for the improvement of the automated PR description generation.

To improve the quality of automatically generated PR descriptions, we propose a novel approach, named PRHAN (**P**ull **R**equests Description Generation Based on **H**ybrid **A**ttention **N**etwork). It can improve the performance from the following three aspects: (1) We propose a novel architecture named hybrid attention network (HAN) and only use HAN to construct our PR description generation model. Compared with RNN-based approaches, HAN can better learn the context information, which helps our model generate high-quality PR descriptions. Moreover, due to replacing RNN architecture, our model has faster execution efficiency. (2) To effectively address the OOV problem, we build an open vocabulary by exploiting byte-pair encoding (BPE) (Sennrich et al., 2016) to the PR dataset. We thus can represent the OOV words by combining sub-word units in the open vocabulary. (3) We introduce the label smoothing (Müller et al., 2019; Vaswani et al., 2017) technology to the cross-entropy loss function, by which our model have stronger generalization.

To evaluate the performance of PRHAN, we choose LeadCM (Liu et al., 2019b), Transformer, and the state-of-the-art approach proposed by Liu et al. (2019b) as baseline models. We conduct a series of experiments on the open source dataset (Liu et al., 2019b) which contains 41K PRs. Especially, we divide the dataset by the project to ensure that we can obtain more realistic results. The experimental results illustrate that our model outperforms all the baselines by 15.38%, 13.81%, and 11.43% under the F1 score of ROUGE and BLEU, respectively. Besides, we also make a human evaluation for all the models, and experimental results show that PRHAN can generate high-quality PR descriptions.

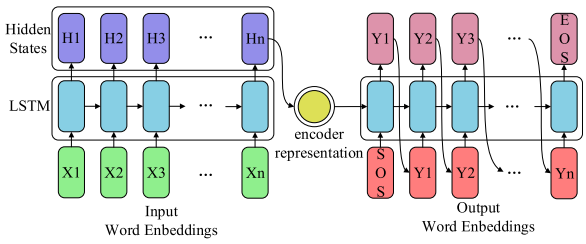

Fig. 2. An example of Seq2Seq model.

To help other researchers quickly reimplement our approach, we make our code available at GitHub[3].

To sum up, the major contributions of our work are as follows:

- To improve the quality of automatically generated PR descriptions, we propose a HAN-based approach, named PRHAN. To the best of our knowledge, we are the first to automatically generate PR descriptions by solely using HAN.
- We build an open vocabulary composed of sub-word units by BPE, to effectively address the OOV problem. Additionally, we also enforce the generalization of PRHAN by introducing label smoothing technology.
- We evaluate the effectiveness of PRHAN on the public dataset through automated metrics and human evaluation. Experimental results demonstrate that PRHAN performs better than all the baselines in terms of effectiveness. Besides, it has faster execution efficiency than the state-of-the-art model proposed by *Liu et al.*.

The remaining of this paper includes the following parts. Section 2 introduces the background knowledge of PRHAN and our motivation. As for Section 3, we elaborate PRHAN in detail, including HAN, BPE, and label smoothing. Sections 4 and 5 present the experimental setup and experimental results, respectively. In Section 6, we make some discussion about the advantages and limitations of PRHAN. As for Section 7, we introduce the related works. Finally, we conclude the paper and point out the future research plans in Section 8.

2. Background and motivation

In this section, we introduce the background of our approach, including seq2seq model and attention mechanism. Afterward, we introduce the motivation of our work.

2.1. Seq2Seq model

The idea of Seq2Seq (Sequence to Sequence) is proposed by Kalchbrenner and Blunsom (2013) who map the entire input sentence into a fixed-length vector and use it to perform machine translation. Afterward, Sutskever et al. (2014) formally constructed the first Seq2Seq model for machine translation task. They used an LSTM model (Hochreiter and Schmidhuber, 1997) to construct an encoder, for mapping the source sentence into a fixed-length vector. Similar to the encoder, they used another LSTM model to construct a decoder, for mapping this fixed-length

[3] https://github.com/TomasAndersonFang/PRHAN

vector into the target sentence (as shown in Fig. 2). All these models promote the development of deep learning (LeCun et al., 2015) in the field of natural language processing (NLP).

Recently, Seq2Seq model has achieved great success in many NLP tasks such as machine translation (Bahdanau et al., 2016; Vaswani et al., 2017), text summarization (Liu and Lapata, 2019), speech recognition (Dong et al., 2018), and question–answering system (Bauer et al., 2018). Our work focuses on automatically generating the high-quality PR description from its commits message. Similar to text summarization, we can regard PR description generation as making the summarization for the commits message of a PR. Therefore, we can take full advantage of Seq2Seq model to design our automated PR description generation model.

2.2. Attention network

2.2.1. Global attention network

As shown in Fig. 3, we perform self-attention network (SAN) (Vaswani et al., 2017) to achieve global attention. For a given PR text $X = \{x_1, x_2, ..., x_n\}$, where n is length of this PR text, we first applies a word embedding for X. Then, we relatively pass the embedding vector pf X to three different linear projections, to obtain queries vector $Q \in \mathbb{R}^{n\times d}$, keys vector $K \in \mathbb{R}^{n\times d}$, and values vector $V \in \mathbb{R}^{n\times d}$, which can be defined as follows:

$$\begin{aligned} Q &= embed(X)\cdot W_Q^T, \\ K &= embed(X)\cdot W_K^T, \\ V &= embed(X)\cdot W_V^T, \end{aligned} \tag{1}$$

where $embed(\cdot)$ denotes the word embedding vector of X, W_Q^T, W_K^T, and W_V^T are the learnable parameters.

After completing the above steps, we obtain the context vector that contains the semantic information of the entire PR by the following steps:

$$context_g = \text{Attn}(Q, K)\cdot V, \tag{2}$$

where $Attn(\cdot)$ is a scaled dot-product attention model, which can be defined by Eq. (3):

$$\begin{aligned} \text{Attn}(Q, K) &= \text{softmax}(attn_g), \\ attn_g &= \frac{Q\cdot K^T}{\sqrt{d}}, \end{aligned} \tag{3}$$

where $\sqrt{d}$ is a temperature factor that can avoid gradient disappearing or explosion of the model during the training phase. The value of d is equal to the feature dimension d. softmax$(\cdot)$ is a normalized function that is used to get the attention weight matrix.

When completing the above steps, we obtain the output of the global attention network. Since SAN is proficient in capturing long-range dependences, the output of the global attention network contains abundant global semantic information that helps the model to learn the long-range dependency (Vaswani et al., 2017).

2.2.2. Local attention

Although SAN can effectively capture long-range dependency, its operation that pays attention to every word in a sentence when it encodes a word may disperse the distribution of attention, which ignores the relation of neighboring words (Yang et al., 2018). Therefore, we introduce local attention (Luong et al., 2015; Yang et al., 2018) which only pays attention to neighboring words. Specifically, we utilize a Gaussian bias $G \in \mathbb{R}^{n\times n}$ to denote local attention (Luong et al., 2015). For each $G_{i,j} \in (-\infty, 0]$, it

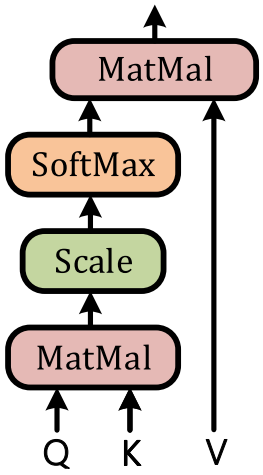

Fig. 3. The structure of self-attention network.

measures the degree of relevance between the word x_i and the central position P_i (we will introduce it later). We define it as:

$$G_{i,j} = -\frac{(j - P_i)^2}{2\sigma_i^2}, \tag{4}$$

where σ_i is the standard deviation. Following the prior work (Luong et al., 2015), we empirically set it as $\sigma_i = \frac{D_i}{2}$, where D_i is a window size. We calculate central position and window size as follows:

$$\begin{bmatrix} P_i \\ D_i \end{bmatrix} = n\cdot \text{sigmoid}\left(\begin{bmatrix} p_i \\ d_i \end{bmatrix}\right). \tag{5}$$

The goal of Eq. (5) is to normalize the max value of P_i and D_i to the length of the input PR sequence. In Eq. (5), p_i and d_i can are calculated as follows:

$$\begin{aligned} p_i &= U_p^T\cdot \tanh(W_p\cdot Q_i), \\ d_i &= U_d^T\cdot \tanh(W_p\cdot Q_i), \end{aligned} \tag{6}$$

For each element in the input sequence, we predict a central position and window size depending on its corresponding query vector Q_i (ith word's query vector in Q), respectively. $U_p \in \mathbb{R}^d$ and $U_d \in \mathbb{R}^d$ are the trainable linear projection. $\tanh(\cdot)$ is an activation function, and W_p is the learnable parameters set.

When completing the above steps, we gain the local attention weight matrix. Note that, because of the exponential operation in softmax function, our local attention matrix $G \in (-\infty, 0]$ is transformed to the attention distribution with a weight $\in (0, 1]$. The output of the local attention network is calculated as follows:

$$context_l = \text{softmax}(G)\cdot V. \tag{7}$$

Different from SAN that can capture long-range dependence, local attention can effectively build the relation of neighboring words, which helps the model to learn the local dependency.

2.3. Motivation

To automatically generate descriptions for PRs, Liu et al. (2019b) proposed the first attentional encoder–decoder model. Their model has achieved the state-of-the-art results at the ROUGE metric. However, when we re-run the source code shared by them, we find that we need to take the two-stage training for their model. Moreover, each training stage requires more than 5 h (with a Tesla V100 GPU) although we only train less than 40,000 data. It means that their model is not suitable to train

Table 1
The comparison between PR descriptions generated by *Liu et al.* and PR descriptions in testing dataset. Note that Avg is the average length of all PR descriptions.

–	Size	Avg
Liu et al.	4392	33.82
Testing dataset	4392	13.81

the large-scale dataset because of expensive time-consuming. However, the scale of the dataset becomes larger because of the rapid development of the software industry. Additionally, we also find that the PR description generated by *Liu et al.* is always shorter than the reference PR description. Additionally, as shown in Table 1, the average length of PR descriptions generated by *Liu et al.* is much shorter than that in the testing dataset. Specifically, there are 2950 (67.2%) PR descriptions in the testing dataset that are longer (at least 1.5 times) than that of generated by *Liu et al.* Although their model achieved the state-of-the-art results at the ROUGE metric, we find that their model performed badly at the BLEU metric. A major reason is that BLEU introduces a brevity penalty to penalize generated texts that are short than their reference texts. Intuitively, an effective model should perform well on all relevant metrics. This fact may imply that *Liu et al.*'s model cannot fully learn the sufficient context information, and thus generate many short PR descriptions. Therefore, we dive into this state-of-the-art model and try to design a more effective model.

Specifically, RNN-based models cannot fully exploit GPUs to accelerate computation. As shown in Fig. 2, in LSTM, the output of the next time step depends on the output of the current time step. To get the output of the encoder, we need to input a sentence into the encoder step by step from left to right. Therefore, RNN-based models cannot achieve parallel computing, which means that GPUs with high performance cannot accelerate RNN-based models. Besides, it is difficult for LSTM to fully learn sufficient context information (Fang et al., 2021; Zhao and Wang, 2019). The purpose of SCST is to use ROUGE metric to directly optimize the model, which can decrease the sensitivity of the model to slight difference between its prediction and the ground truth (Liu et al., 2019b). However, SCST requires a two-stage training mode, which doubles the time-consuming in the training phase. To train their model, Liu et al. (2019b) built a close vocabulary with 50,000 unique words that have the highest frequency and integrated the pointer network to address the OOV problem, which makes their model complicated because the pointer network introduces a large number of parameters.

Motivated by the above-mentioned problems, we actively explore how to deal with them. In this process, we note that Transformer (Vaswani et al., 2017) is designed by SAN, which enables it to achieve parallel computing and learn context information. Thus it has achieved great success in all kinds of NLP tasks (Devlin et al., 2019; Vaswani et al., 2017; Yu et al., 2018; Zhang et al., 2019). Inspired by this novel architecture, we start to explore how to only use the attention network to construct a suitable model for PR dataset. In addition, to control the size of the vocabulary and represent the OOV words, we premeditate to adopt the sub-word encoding techniques (Devlin et al., 2019; Sennrich et al., 2016) to PR dataset. Finally, we introduce the label smoothing (Müller et al., 2019; Vaswani et al., 2017) technology to our model, for reducing the sensitivity of neural network models to slight differences and enforce its generalization.

Based on the above exploration, in the phase of data preprocessing, we apply BPE to the PR dataset and construct a small-size vocabulary (about 5000 unique words) at the sub-word level. Thus we can represent the OOV words by combining sub-word units. Then we directly utilize HAN to construct our model PRHAN which took into account both effectiveness and efficiency, to automatically generate PR description for each PR. Since HAN is the combination of SAN and the local attention network, it learns sufficient context information and can achieve parallel computing. Finally, we optimize PRHAN by adding label smoothing to the cross-entropy loss function, which can reducing the sensitivity of neural network models to slight differences and enforce its generalization. Moreover, label smoothing does not complicate our model.

3. Approach

Attention-based approaches have achieved great success in many NLP tasks (Devlin et al., 2019; Lan et al., 2020; Li et al., 2019; Vaswani et al., 2017; Veličković et al., 2018; Yu et al., 2018; Zhang et al., 2019). Inspired by them, we build HAN by combining the global attention and local attention networks, then utilize HAN to construct PRHAN. In this section, we describe PRHAN in detail, including HAN, BPE, and label smoothing.

3.1. Byte pair encoding

Byte pair encoding, proposed by Gage (1994), is a compression algorithm that can combine the most frequent pair of bytes at each iteration. Sennrich et al. (2016) applied BPE to neural machine translation since it effectively alleviates the OOV problem. Specifically, BPE can represent an open vocabulary by a fixed-length vocabulary with variable-length character sequences. In other words, BPE conducts the sub-word encoding for the initial vocabulary and transforms words to sub-word units. Therefore, although a word is not in the vocabulary, it is can be represented by combining sub-word units. BPE thus takes large improvement to many NLP tasks, such as machine translation (Gehring et al., 2017; Vaswani et al., 2017; Wu et al., 2016) and text classification (Conneau et al., 2017; Liu et al., 2019a).

Here, we describe the implementation process of BPE by introducing a simple toy example. Suppose that there is a tiny corpus which only contains four words, we can use a dictionary to count these words and their frequencies, which is presented as follows:

— {'low' : 5, 'lower' : 2, 'newest' : 6, 'widest' : 3}.

Firstly, we decompose each word to the minimum units and get the initial dictionary:

— {'l', 'o', 'w', 'e', 'r', 'n', 'w', 's', 't', 'i', 'd'}.

Secondly, we pair the minimum units from left to right for each word, *i.e.*, ('l', 'o') and ('o', 'w') for word "low". According to the dictionary, we note that the token pair ('e', 's') has the highest frequency (9 times). Hence, we combine the minimum units 'e' and 's' and get a new sub-word 'es'. Then, we add the new sub-word 'es' to the initial vocabulary and remove the minimum unit 's' because no other sub-words contain it.

— {'l', 'o', 'w', 'e', 'es', 'r', 'n', 'w', 't', 'i', 'd'}.

Finally, we continue repeating the above step until the frequency of each adjacent minimum unit pair equals to 1. Note that the vocabulary size in our example is tiny, thus it is difficult to show the process of compressing the vocabulary well. When finishing BPE for our corpus, we get a new vocabulary composing of sub-word units, including 'low_', 'er', 'new_', 'wid_', and 'est'. Therefore, although the word "lowest" is not in our corpus, it can be represented by combining sub-words 'low_' and 'est'.

3.2. Hybrid attention network

To leverage the global and local semantic information from the global and local attention networks, we construct a hybrid attention network by merging these two attention network. Generally, we can directly use addition operation to achieve our goal. However, this operation cannot balance the global and local attention (Xu et al., 2019a). In other words, our model cannot effectively take advantage of these two attention networks. Therefore, we introduce a learnable gated scalar α to achieve dynamic balance, which is represented as:

$$attn = (1 - \alpha) \times attn_g + \alpha \times G, \tag{8}$$

where α can be calculated as follows:

$$\alpha = \sigma(embed(X) \cdot W_g^T), \tag{9}$$

where σ is the sigmoid activation function, $embed(X)$ is described in Section 2.2.1, and W_g^T is learnable parameter set. Therefore, the output of HAN is computed as follows,

$$context = \text{softmax}(attn) \cdot V. \tag{10}$$

Here, *context* vector contains abundant global and local semantic information. PRHAN thus can learn sufficient context information used to generate high-quality PR descriptions.

3.3. Label smoothing

In the process of training, neural networks tend to become too confident about their prediction (Müller et al., 2019). In the PR description generation task, this phenomenon is reflected in the fact that the cross-entropy loss function gives punishments to slight differences between the prediction and ground truth. Unfortunately, this fact influences the performance and generalization of the model.

To alleviate this problem, Liu et al. (2019b) integrated SCST, a reinforcement learning technology that enables them to directly utilize ROUGE metric to optimize their model. Therefore, their model can generate valid PR descriptions. However, such optimization mode requires a two-stage training, which complicates their model and increases their training cost. Moreover, although SCST makes their model obtain a high ROUGE score, we find their model performs badly on the BLEU metric. Therefore, we choose another simple but effective and objective method, named label smoothing. It has been proved to be an effective method for training neural networks (Müller et al., 2019; Vaswani et al., 2017). It can effectively improve the accuracy of image classification (Müller et al., 2019; Gao et al., 2020), machine translation (Vaswani et al., 2017), and speech recognition (Wang et al., 2019).

In short, label smoothing makes neural networks become a little humble and keep suspicious of the "correct answer" by adding it to the cross-entropy loss function. Thus, PRHAN is encouraged to generate diverse PR descriptions, which reduces the sensitivity of the cross-entropy loss function as well as enforces the generalization of PRHAN. Specially, label smoothing does not complicate our model since it is only an extra unit to the cross-entropy loss function. In Section 3.4, we further describe how to add the label smoothing to the cross-entropy loss function.

3.4. Training

In the training phase, we can calculate the vocabulary distribution as follows:

$$P_v = \text{softmax}(context \cdot W_v^T), \tag{11}$$

where *context* is the output of HAN and W_v^T is a learnable parameter. Then we can calculate the jth predicted token $\hat{y_j}$ as follows:

$$P_v(\hat{y_j}) = p(\hat{y_j}|\hat{y_0}, \ldots, \hat{y_{j-1}}, w), \tag{12}$$

where $\hat{y}$ is the input of the decoder and w is the output of the encoder. Next, we optimize our model by minimizing the revised cross-entropy loss function, as follows:

$$\mathcal{L}(\theta) = -\sum_{j=1}^{n} \log(P_v(\hat{y_j})) \times P_r(y_i), \tag{13}$$

where θ is the trainable parameter set and y_i is the ground-truth token in the vocabulary. $P_r(y_j)$ is a mixture of original ground-truth distribution $P_o(y_j)$ and a random distribution $U(y_i)$ (as noise). We can be defined $P_r(y_j)$ as:

$$P_r(y_j) = (1 - \epsilon) \times P_o(y_j) + \epsilon \times U(y_i), \tag{14}$$

where ϵ is the label smoothing parameter. In our experiments, we set it as 0.1. By introducing label smoothing, we encourage PRHAN to generate more fluent results and have stronger generalization. Therefore, we reduce the sensitivity of the cross-entropy loss function and improve the generalization ability of PRHAN. Besides, we use Adam optimizer (Kingma and Ba, 2017) to optimize the revised cross-entropy loss function.

The whole workflow of PRHAN is shown in Fig. 4. We first utilize BPE to build a vocabulary with 2500 unique sub-word units. Then we use this new vocabulary to generate a new dataset composed of the sub-word units. Afterward, we use the new generated dataset to train our model by minimizing the revised cross-entropy loss function. After finishing training, we can generate PR description for every PR by inputting it to our trained model.

4. Experiment setup

4.1. Dataset and baselines

4.1.1. Dataset

We conduct experiments on the open source dataset shared by Liu et al. (2019b). Specifically, they collected more than 20,000 repositories which at least contains one merged PR. Then they download the top 1000 projects according to the number of merged PRs in these repositories. In the literature (Liu et al., 2019b), the authors randomly choose 10% of data in the dataset for validation and testing, respectively, and the remaining data for training, which is the same as Jiang et al. (2017) and Hu et al. (2018). However, according to *LeClair et al.*'s study (LeClair and McMillan, 2019), they find that the code methods from the same project are more possible to have similarities. Therefore, if we divide the dataset by the above way, PR in the same project may be in training dataset and testing dataset simultaneously, which may make us obtain wrong evaluation results. Following *LeClair et al.*'s study, we divide the dataset by the project. As shown in Table 2, we randomly choose 10% of projects in the dataset for validation and testing, respectively, then use the remaining projects to train the model. Since we divide the dataset by the project, there is a little difference between the size of the validation dataset and the testing dataset. Following the prior work (Liu et al., 2019b), we first apply basic pre-processing, such as filtering non-English and special symbols, tokenizing, and removing *non-ASCII* tokens, to the data. After completing this step, we utilize BPE to perform compression operation for our pre-processed data from 694 projects. Then, we generate a new dataset composed of sub-word units as well as a vocabulary that contains about 2500 unique sub-word units (Karampatsis et al., 2020). A PR example after pre-processing is shown in Fig. 5.

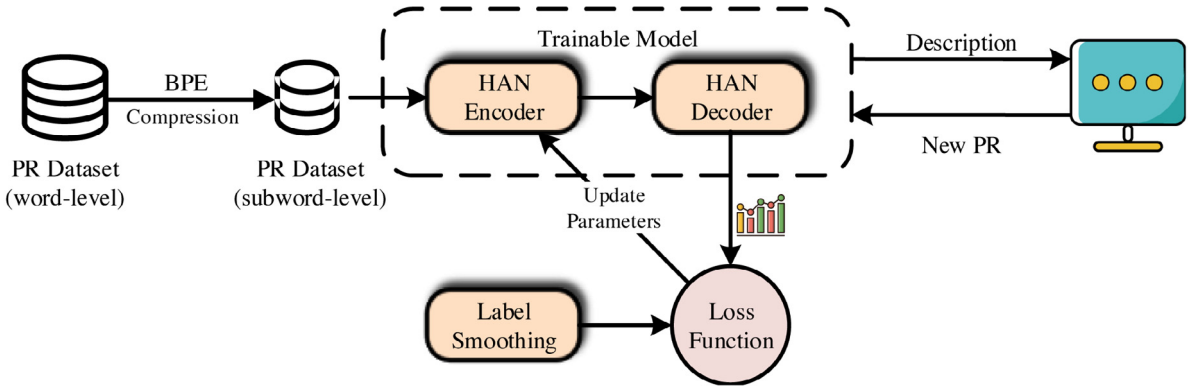

Fig. 4. The whole workflow of the model.

Table 2
Statistics of the dataset. "Project" denotes the number of different projects in the set and "Size" is the number of pairs of pull requests and descriptions in the set.

-	Project	Size	Language
Training set	555	33,430	Java
Validation set	69	4,010	Java
Testing set	70	4,392	Java

PR Sequence:
implemented store ' result ' field using open@@ m@@ rs ob@@ s <nl> note : not working version . just for pe@@ er review .
<cm-sep> implemented store ' result ' field using ob@@ s group@@ ing <nl> note : fail when setting members .
<cm-sep> implemented retriev@@ ing ' result ' in ob@@ s group <nl> note : not working solution . conf@@ using behaviour .
retrie@@ ved ob@@ s be@@ have de@@ feren@@ tly form created one . <cm-sep> fixed the issue un@@ able to store using ob@@ s group@@ ing .
<cm-sep> merge branch ' master ' into f@@ m-0 .
<cm-sep> removed extra logs .

Reference: is it necessary to include ' result ' in the ' contained ' field

Fig. 5. A PR that has been processed. The "@@" denotes connection symbol.

4.1.2. Baselines

LeadCM (Liu et al., 2019b): For each PR sequence, LeadCM outputs the first sentence of its commit messages as the corresponding description. In most cases, the description generated by LeadCM is contained in the first commit message of this PR. The reason is that developers are accustomed to putting the key changes in the first commit, which may contain the core changes of the PR (Liu et al., 2019b). As the first sentence in the first commit, it is usually a summarization for the commit. In our experiments, we reimplement LeadCM.

Transformer (Vaswani et al., 2017) has achieved state-of-the-art results in all kinds of tasks in NLP (Lee et al., 2020; Vaswani et al., 2017; Yu et al., 2018). Ahmad et al. (2020) designed a Transformer-based model for the code summarization task and achieved good results on the Java dataset shared by Hu et al. (2018). Note that *Ahmad et al.* added relative position encoding (Shaw et al., 2018) and pointer generator (See et al., 2017) into the basic Transformer, which can enrich local context information and alleviate the OOV problem. In our experiments, we use the original Transformer because it is more representative. In our experiments, we implement the basic Transformer by re-running the source code shared by Vaswani et al. (2017).

Liu et al.'s (Liu et al., 2019b) **proposed model:** Aiming at the task of automatically generating PR description, they proposed a novel seq2seq model that integrates the pointer network (See et al., 2017), attention mechanism (Bahdanau et al., 2016; Luong et al., 2015), and SCST (Rennie et al., 2017). In our experiments, we re-run their shared source code[4] on the dataset divided by the project.

4.2. Evaluation metrics

Following the prior PR summary task (Liu et al., 2019b), we use the ROUGE metric to evaluate our model. ROUGE[5] is the most used metric in text summarization because it is highly correlated with human evaluation (Fernandes et al., 2021; Gehrmann et al., 2018; Lebanoff et al., 2019; Li et al., 2017). We mainly use ROUGE-1, ROUGE-2, and ROUGE-L to evaluate all the models used in our experiments. For each metric, we calculate Recall, Precision, and F1-score, respectively.

Specifically, we calculate F1 score for ROUGE-N as follows:

$$F1_{rouge-n} = \frac{2R_{rouge-n}P_{rouge-n}}{R_{rouge-n} + P_{rouge-n}}, \tag{15}$$

where $R_{rouge-n}$ and $P_{rouge-n}$ are the recall and precision for ROUGE-N:

$$\begin{aligned} R_{rouge-n} &= \frac{\sum_{(gen,ref)\in S}\sum_{gram_n\in ref} Cnt_{gen}(gram_n)}{sum_{(gen,ref)\in S}\sum_{gram_n\in ref} Cnt_{ref}(gram_n)}, \\ R_{rouge-n} &= \frac{\sum_{(gen,ref)\in S}\sum_{gram_n\in ref} Cnt_{gen}(gram_n)}{sum_{(gen,ref)\in S}\sum_{gram_n\in gen} Cnt_{gen}(gram_n)}, \end{aligned} \tag{16}$$

where *gen*, *ref*, and *S* denote the PR description generated by our models, reference description, and test dataset, respectively. $gram_n$ is the n-gram phrase. $Cnt_{gen}(gram_n)$ and $Cnt_{ref}(gram_n)$ denote the frequency that $gram_n$ occurs in *gen* and *ref*, respectively. In general, ROUGE is usually expressed as a percentage value between 0 and 100.

Except for using ROUGE to measure the quality of generated PR descriptions, we also use BLEU to measure their quality. The reason is that BLEU is widely used in code summarization

[4] https://github.com/Tbabm/PRSummarizer.
[5] https://github.com/bheinzerling/pyrouge.

Table 3
Parameters settings for training.

Name	Description	Value
GPU	Tesla V100	32 GB
CPU	Xeon(R) E5–2698	512 GB
d	Feature dimension	128
b_t	Batch size	128
Layers	Layers of HAN	3
Dropout	–	0.1
ϵ	Label smoothing	0.1
lr	Learning rate	0.0005
epoch	Number of iterations	100

task (Ahmad et al., 2020; Liu et al., 2021; Wei et al., 2020) and PR description generation task is similar to code summarization task. Moreover, an effective model should perform well on all relevant metrics. The specific definition of BLEU metric is presented as follows:

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n), \tag{17}$$

where BP is a brevity penalty that avoids a high score of shorter generated comment, w_n is a weight factor which is empirically set to $1/N$ (Papineni et al., 2002), and p_n is the geometric average of the modified n-gram precision. According to the previous studies (Hu et al., 2018; LeClair et al., 2019), we set N to 4. BLEU score ranges from 0 to 100, which indicates that the higher BLEU score is, the more accurate and fluent the generated PR description is.

To comprehensively measure the effectiveness of the model, we extra calculate the F1-score for ROUGE and BLEU, which is presented as follows:

$$\begin{aligned} F1-RB &= \frac{2*(ROUGE*BLEU)}{(ROUGE+BLEU)}, \\ ROUGE &= \frac{(F1_{rouge-1}+F1_{rouge-2}+F1_{rouge-l})}{3}, \end{aligned} \tag{18}$$

where $F1-RB$ expresses the F1-score for ROGUE and BLEU.

4.3. Experiment settings

At the training phase, we train our model for 120 epochs with the revised cross-entropy loss function, and evaluate our model every epoch and save the best one. The specific parameters setting is shown in Table 3.

At the inference phase, we leverage beam search algorithm with size of 4 to generate PR description sequence, which is same as Liu et al. (2019b). Beam search algorithm makes great success in many software engineering and NLP tasks, such as code to comment (Ahmad et al., 2020), program repair (Lutellier et al., 2020), machine translation (Vaswani et al., 2017), and text summarization (Liu and Lapata, 2019).

4.4. Research questions

Our work focuses on the following three research questions (RQ):

> **RQ1:** Compared to baselines, can PRHAN generate high-quality PR descriptions?

Motivation: RQ1 is designed to verify whether PRHAN can generate the high-quality natural language descriptions for PRs. To achieve this goal, we utilize the baselines and PRHAN to generate PR descriptions for PRs, then we measure the quality of generated PR descriptions by ROUGE, BLEU, and F1-RB metrics.

Approach: We choose three models, including LeadCM, Transformer, and the state-of-the-art model proposed by Liu et al. (2019b), as baselines. Then we evaluate the performance of these baseline models and PRHAN by measuring the quality of PR descriptions generated by them. In our experiments, we use ROUGE-1, ROUGE-2, ROUGE-L, BLEU, and F1-RB to measure all the generated PR descriptions. Therefore, an effective model should perform well in all the metrics.

> **RQ2**: Can PRHAN get good efficiency for generating PR descriptions?

Motivation: RQ2 aims at exploring the model's efficiency. Although the state-of-the-art model (Liu et al., 2019b) can generate valid PR descriptions, it needs a huge amount of time in training and a long waiting time in generating a PR description. As opposed to their model, our model PRHAN is solely based on HAN, which has a high execution efficiency. Therefore, we build a group of experiments to verify the efficiency of PRHAN.
Approach: Firstly, we count the total parameters of two models. Next, we compute the training time and generation time for each sample, which can directly reflect the efficiency of the model.

> **RQ3**: How is the effectiveness of every component in the whole approach?

Motivation: As described in Section 3, PRHAN is designed based on HAN. In addition, we introduce two independent methods that include BPE and label smoothing, to further improve the performance of PRHAN. BPE aims at the OOV problem and vocabulary size, and label smoothing reduces the sensitivity of PRHAN. These two methods all can boost the efficiency and effectiveness. Therefore, the goal of RQ3 is to verify the validity of each component in PRHAN.
Approach: To evaluate the effectiveness of each component, we design two extra models, i.e., HAN+BPE and HAN. HAN+BPE denotes that we remove label smoothing in PRHAN and HAN+BPE denotes that we only remove label smoothing in PRHAN. Similar to RQ1, we use ROUGE and BLEU metrics to evaluate these models and verify the effectiveness of each component.

5. Experimental results

5.1. RQ1: Model effectiveness

Results: Table 4 shows the performance of the baseline models and HANPR under the metrics of ROUGE-1, ROUGE-2, ROUGE-L, BLEU, and F1-RB. Especially, for the ROUGE metric, we calculate the F1 score for it since the F1 score can comprehensively reflect the effectiveness of the model. For the metric of ROUGE-1, HANPR outperforms LeadCM, Transformer, and *liu et al.*'s (Liu et al., 2019b) model by 39.43%, 94.67% and 11.05%, respectively. For the ROUGE-2 metric, HANPR outperforms LeadCM, Transformer, and *liu et al.*'s (Liu et al., 2019b) model by 118.11%, 193.15% and 30.58%, respectively. For the metric of ROUGE-L, HANPR has the 65.90%, 73.65% and 17.53% performance improvement compared to LeadCM, Transformer, and *liu et al.*'s (Liu et al., 2019b) model, respectively. This metric shows that compared to the three baselines, HANPR can learn the sufficient and key context information of a PR, and it thus can generate a high-quality PR description. To comprehensively evaluate all models, we extra calculate the BLEU score and F1-RB score for them. For the BLEU score, we can note that PRHAN outperforms all baselines by large margins (from 10.89 to 13.63). The reason is that BLEU introduces brevity penalty to penalize the generated

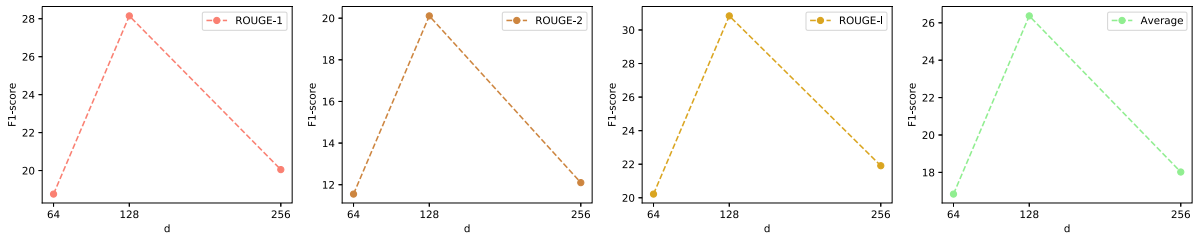

Fig. 6. The influence to PRHAN when we choose different feature dimension d (we set d to 64, 128, and 256, respectively).

Table 4
Performance comparison between HANPR and baseline models in term of F1-score under the metrics of ROUGE-1/2/L, BLEU score, and F1-RB score.

Model	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	F1-RB
LeadCM	20.19	9.22	18.59	4.07	6.49
Transformer	14.46	6.86	17.76	5.83	8.06
liu et al.'s	25.35	15.40	26.24	6.81	10.44
HANPR	**28.15**	**20.11**	**30.84**	**17.70**	**21.87**

Table 5
Comparison of the efficiency between the state-of-the-art model proposed by *Liu et al.* and PRHAN. Note that the parameters denote the total model parameters and memory denotes GPU memory.

Model	Parameters	Memory	Training
liu et al.'s	27.90M	31 GB	11.317h
PRHAN	1.70M	11 GB	1.458h

Table 6
Effectiveness of each component.

Model	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	F1-RB
HAN	18.10	9.15	20.30	8.12	10.74
HAN+BPE	26.62	19.05	29.13	16.31	19.72
PRHAN	**28.15**	**20.11**	**30.84**	**17.70**	**21.87**

text that is shorter than the reference text. Therefore, BLEU is a stricter metric that requires the model to generate more accurate PR descriptions, which means that the model needs to better learn sufficient context information from PR. Compared to baseline models, PRHAN combines two attention networks, which make it fully take advantage of BPE and learn sufficient context information from PRs. As for the F1-RB metric, because of the low BLEU score of baseline models, PRHAN outperforms them by large margins (from 11.43 to 15.38). Considering that an effective model should perform well on different relevant metrics, the above experimental results indicate that PRHAN is a more effective model for automated PR description generation.

Except for evaluating baselines and PRHAN on ROUGE, BLEU, and F1-RB metrics, we also count the length of PR descriptions generated by PRHAN and *Liu et al.*'s model. According to our investigation, 81.6% of PR descriptions generated by *Liu et al.* are shorter than the reference PR descriptions, and 55.6% of their PR descriptions are half of the length of the reference PR descriptions. As for PR descriptions generated by PRHAN, 48.1% of them are shorter than the reference PR descriptions, and only 19.3% of them are half of the length of the reference PR descriptions. Considering that PRHAN obtains higher scores on all metrics, thus PRHAN can generate high-quality PR descriptions.

> **Answer to RQ1**: PRHAN performs better than all baseline models at metrics of ROUGE-1, ROUGE-2, ROUGE-L, BLEU, and F1-RB, thus it is an effective model.

5.2. RQ2: Model efficiency

Results: In Table 5, we can note that the number of parameters of *Liu et al.*'s model (Liu et al., 2019b) is 16 times that of PRHAN. Therefore, their model requires more memory (20 GB) than PRHAN under the same batch size. In fact, re-running their model almost uses all memory of our GPU. To compare model efficiency fairly, we train these two models in the same experimental environment. The experimental results demonstrate that *Liu et al.*'s (Liu et al., 2019b) model takes 11.317 h to train, while PRHAN only spends 1.458 h on training. PRHAN can save about 10 h of training time. Considering that the dataset we use is small and the scale of the dataset becomes larger because of the rapid development of the software industry, it means that HANPR executes faster than the state-of-the-art model and is more suitable to use in practice. Although we find that *Liu et al.*'s model (Liu et al., 2019b) becomes faster if we use a smaller batch size, such as 8, it is not suitable for the large scale dataset because it cannot make full use of the GPU memory and increases the number of iterations.

> **Answer to RQ2**: Comparing with the state-of-the-art model, PRHAN has the significant enhancement in efficiency.

5.3. RQ3: Ablation experiment

Results: PRHAN is based on seq2seq model and HAN. To control the size of vocabulary and cope with the OOV problem, we introduce BPE. Additionally, we add label smoothing to the cross-entropy loss function to reduce the sensitivity of our model as well as enforce its generalization.

As shown in Table 6, when we utilize BPE to deal with the vocabulary, HAN+BPE outperforms HAN by 47.07%, 108.20%, 43.50%, 100.86%, and 83.61% in terms of ROUGE-1, ROUGE-2, ROUGE-L, BLEU, and F1-RB, respectively. This means that BPE is helpful for improving the performance by using the sub-word units to represent the OOV words, especially for ROUGE-2 score. The reason is that the OOV words are very harmful to the ROUGE-2 metric because they can break the continuity of words. The high improvement for the BLEU score demonstrates that BPE can help the model generate more fluent PR descriptions with a close length to the reference PR descriptions. We also can observe that the performance of HAN+BPE can be further improved by introducing label smoothing. When adding the label smoothing, it brings improvement to HAN+BPE by 5.75%, 5.56%, 5.87%, 8.52%, and 19.90% in terms of ROUGE-1, ROUGE-2, ROUGE-L, BLEU, and F1-RB, respectively. Therefore, the label smoothing can effectively reduce the sensitivity of the model and make it generate high-quality PR descriptions. Especially, using label smoothing cannot add extra parameters to the model, which means that it does not increase the training time.

Moreover, we extra test the influence on the performance of PRHAN when we choose different parameters for each component. For HAN, we mainly evaluate its influence on PRHAN when

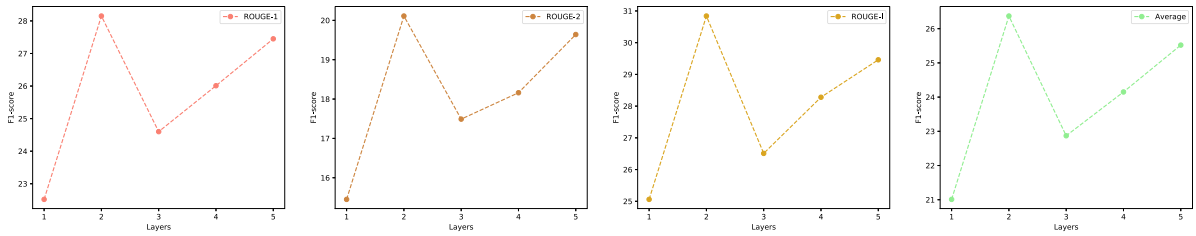

Fig. 7. The influence to PRHAN when we choose different layers of HAN.

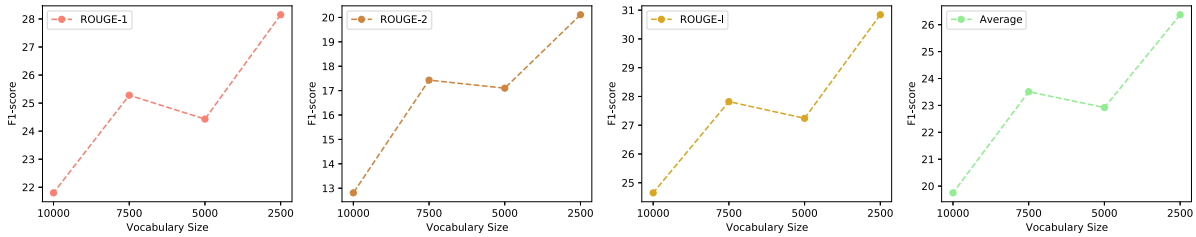

Fig. 8. The influence to PRHAN when we choose vocabulary with different size (we set vocabulary size to 2500, 5000, 7500, and 10,000, respectively).

we stack the different numbers of HAN layers or select feature dimension d. For BPE, we use it to generate multiple vocabularies with different sizes and estimate their effects on our model.

As shown in Fig. 6, these four pictures from left to right are F1-score of PRHAN under the metrics of ROUGE-1, ROUGE-2, ROUGE-L, and the average score of these three metrics, respectively (this setting is also used in Figs. 7 and 8). We observe that F1-score keeps increasing until we set $d = 128$ in term of ROUGE-1, ROUGE-2, ROUGE-L, and the Average. Thus, PRHAN achieves the best performance when we set d to 128. In Fig. 7, we can distinctly find that when we stack two HAN layers, PRHAN obtains the best F1-score in terms of ROUGE-1, ROUGE-2, ROUGE-L, and the Average. Although stacking five HAN layers enables PRHAN to obtain the close result, PRHAN with two HAN layers has fewer parameters than that with five HAN layers. Therefore, PRHAN built by two HAN layers with feature dimension $d = 128$ can achieve state-of-the-art results. In Fig. 8, we can observe that when we set the vocabulary size to 2500, PRHAN gets a significant improvement in terms of ROUGE-1, ROUGE-2, ROUGE-L, and the average. Therefore, setting vocabulary size to 2500 is a good choice.

> **Answer to RQ3:** To sum up, the experimental results support the effectiveness of each component in PRHAN, which demonstrates the possibility that applying these components to other software engineering tasks, i.e., code search (Fang et al., 2021; Gu et al., 2018) and code summarization (Hu et al., 2018; Iyer et al., 2016). In addition, we find when we choose a two-layer HAN with feature dimension $d = 128$ and set vocabulary size to 2500, PRHAN can obtain the best performance.

6. Discussion

6.1. Why does our proposed model work well?

As described in Section 3.2, HAN is composed of the global attention network and the local attention network. Thus, HAN-based model can learn both the global information and local information of a PR, which helps the model to deeply understand the PR and generate a suitable description for it. Although HAN can learn sufficient context information, there is serious OOV problem in PRs, which limits the ability of HAN to learn the context information from PRs. For example, in the testing dataset, more than 80% of PRs have at least one OOV token, and about 25% of PRs have more than five OOV tokens. We alleviate the OOV problem by introducing sub-word encoding, i.e., BPE. By using BPE to build an open vocabulary at the sub-word level, we can represent words by combining sub-word units in the vocabulary. Although some words in the testing dataset are not in the training dataset, we can also represent them by combining sub-word units in the vocabulary. For example, when using sub-word vocabulary to represent PRs in the testing dataset, the OOV problem vanishes. Therefore, BPE can effectively solve the OOV problem, which liberates the potential of HAN. Additionally, BPE also can reduce the vocabulary size because the number of sub-word units are far less than the number of words. From Figs. 9 and 10, we note that compared to baselines, PRHAN can generate a high-quality PR description because HAN learns more sufficient context information from the PR which is in form of the sub-word.

6.2. Human evaluation

In the above experiments, PRHAN achieves the state-of-the-art results on the ROUGE and BLEU metrics. However, both ROUGE and BLEU are automatic metrics and literature (Stapleton et al., 2020) indicates that there is no obvious evidence to support the correlation between automatic metrics and human evaluation on comprehension and coding tasks. Therefore, it is necessary to conduct a human evaluation to support the effectiveness of PRHAN. Following the prior studies (Iyer et al., 2016; Liu et al., 2019b; Stapleton et al., 2020; Wei et al., 2020), we invite ten human evaluators to give quality scores to the PR descriptions generated by our approach and baselines. The invitees include one Ph.D. student, three master students, and six developers, every of whom has 3–5 years of experience in Java programming and is proficient in English writing. We ask every evaluator to compare the PR descriptions generated by all concerned models with the referencing PR descriptions and measure the quality of generated PR descriptions by the following three aspects:

(1) The similarity of generated PR descriptions and references.
(2) The naturalness of generated PR descriptions (grammar and fluency).

storm-0 : mask the plaintext passwords from the logs <nl> introduce a ' password ' config annotation and use it to mark configs that are <nl> sensitive and mask the values while logging .
<cm-sep> storm-0 : remove guava dependency . <cm-sep> revert ' storm-0 : remove guava dependency ' <nl> this reverts commit sha .
<cm-sep> storm-0 : use shaded guava and remove the usage of redactvalue <nl> change-id : idb0775f1c1b91cbc648087e7752783f119e034b3 . <para-sep> ignore .

Reference: introduce a ' password ' config annotation and use it to mark configs that are <nl> sensitive and mask the values while logging .

LeadCM: storm-0 : mask the plaintext passwords from the logs <nl> introduce a ' password ' config annotation and use it to mark configs that are <nl> sensitive and mask the values while logging .

Transformer: introduce a ' password password config config and use it to highlight configs that are < nl > processing and the values were logging when logging logging logging .

Liu et al.: introduce a ' password ' config annotation and use it to mark configs that mask the values while logging .

PRHAN: introduce a ' password ' config annotation and use it to mark configs that are < nl > sensitive and mask the values while logging .

Fig. 9. An example of generated PR descriptions.

applied needscdmunittest where necessary . <nl> * also , the travis build matrix now only includes ' oraclejdk8 ' .
<cm-sep> fixed more tests on version <nl> * added needscdmunittest to one class . <nl> * testremotecatalogrequest no longer extends testcase . fixes ' no tests <nl> found ' error .

Reference: - also , the travis build matrix now only includes ' oraclejdk8 ' .

LeadCM: applied needscdmunittest where necessary .

Transformer: this pr fixes tests for tests and tests pass <unk> tests . < nl > tests pass tests for tests pass <unk> tests and tests tests . < nl > added tests tests for <unk> tests <unk> tests <unk> tests .

Liu et al.: also , the travis build matrix now only includes ' oraclejdk8 '

PRHAN: - also , the travis build matrix now only includes ' oraclejdk8 ' .

Fig. 10. An example of generated PR descriptions.

Table 7
Human evaluation for baseline models and PRHAN.

Model	Human score
Leadcm	3.46
Transformer	2.83
Liu et al.	4.67
PRHAN	5.58

(3) The informativeness of generated PR descriptions (the amount of contents carried over from the input PR to the generated description when ignoring its fluency).

Specifically, we randomly choose 50 PRs from the testing dataset, then use our approach and baselines to generate PR descriptions for them. Then, for each group of PR descriptions generated for a PR, we show these PR descriptions to evaluators by the random order. For example, for the first PR description in a group, it is possible to be generated by LeadCM, Transformer, *Liu et al.*'s model, or HANPR, which makes sure that evaluators do not know which output is generated by our model. Note that we generate ten questionnaires by independently conducting ten times process that generates questionnaires, thus each of them is different. Each evaluator should provide the quality score, from 0 to 7, for each PR description generated by each model according to the reference PR description. A higher quality score means that the generated PR description is similar to the reference, is full of naturalness, and has enriched informativeness. To further keep fair, we remove the highest and lowest scores in ten quality scores of each PR description given by evaluators. Therefore, each PR description generated by a model obtains eight quality scores and we calculate the average score as its final score.

Results: The results are shown in Table 7, from the table we can observe that PRHAN obtains the highest human score, which is the same as the results of the automatic measure. This fact further supports the conclusion that PRHAN can generate high-quality PR descriptions than the baselines. However, we also find that PRHAN had not performed well in the special case, we introduce it in detail in the next section.

6.3. When does PRHAN fail to perform well?

Although PRHAN can generate high-quality descriptions for PRs in our dataset. We also need to realize that many PRs in our dataset contains sufficient text information, which is the reason that PRHAN can perform well. However, we find that there are some PRs in dataset without sufficient text information or even no text information. In this case, we find that PRHAN had not performed well both in automatic measures and human evaluation. As shown in Figs. 11 and 12, PRHAN cannot generate low-quality descriptions for given PRs since these two PRs lack enough text information. The major reason is that we use PRs that contain sufficient text information to train PRHAN, and it thus cannot generate valid descriptions for the PRs without sufficient text information. Inspired by *Zhang et al.*'s (Zhang et al., 2017) study about bug report enrichment, we also can enrich PR without sufficient or high-quality text information. For example, we observe that a PR usually is related to the source code, thus PR may be enriched by introducing the semantic information of the source code, which can alleviate the lack of text information in PRs. We will study it in the future work.

6.4. Threats to validity

6.4.1. Internal threats

In our view, the internal threats to our proposed model focus on two aspects. The first threat is that the experimental environment we use is different from which Liu et al. (2019b) used. To

added cloudevent domain for /v2/events . <cm-sep> fixed formatting .

Reference: fixed formatting and squashed commits as requested for pr 0

PRHAN: added cloudevent domain for /v2/events

Fig. 11. An example of bad PR description generated by PRHAN.

tests to verify login page reload based on csrf token expiry . <cm-sep> reload login page when csrf token expires .

Reference: when login page is left open until the point of expiry of csrf token and the user goes on to submit their credentials , they are shown with the csrf error ' invalid login attempt , request does not meet our security standards , please try again ' . this can be avoided by performing a client-side page reload using ' meta-refresh ' when the csrf token expires .

PRHAN: reload login page when csrf token expires < nl > yes < nl > - ran findsecuritybugs plugin and verified report yes < nl > - confirmed that this pr does n't commit any keys , pass

Fig. 12. An example of bad PR description generated by PRHAN.

solve this threat, we evaluate all the models on our server. However, if we replicate baseline models by ourselves, its performance may have deviations to some extend. To mitigate this threat, we re-ran baseline models with the source code shared by Liu et al. (2019b). The second threat is the occasionality of an experimental result. Because of the complexity of the neural networks, a single experimental result cannot fully present the performance of the model. To mitigate this threat, we repeat each experiment five times and average the experimental results.

6.4.2. External threats

The external threat to this work is that the scale of the dataset in our experiments is small. Although PRHAN performs well in the test set, it cannot prove that it is applicable to the large-scale dataset. We plan to collect a more large-scale dataset to further check the generalization ability of our proposed model.

7. Related work

In this section, we describe some related studies, including summarizing software changes, summarizing other software artifacts, and attention mechanism.

7.1. Automated description generation for software changes

Commits, PRs, and releases are software modifications with distinct granularity. A commit is an individual change to a file or a set of files. A PR is the change to a repository, which usually contains multiple commits. As for release, it is a deployable software iteration that can be packaged and made available for a wider audience to download and use. To automatically generate commit messages summarizations, some tools and methods have been proposed (Cortés-Coy et al., 2014; Huang et al., 2017; Jiang et al., 2017; Loyola et al., 2017; Nie et al., 2020; Xu et al., 2019b). For example, Cortés-Coy et al. (2014) proposed ChangeScribe which first recognized the stereotype of modified methods in a commit by introducing abstract syntax tree. Then they utilized pre-defined filters and templates to generate a description for this commit. Similarly, Xu et al. (2019b) constructed CoDiSum based on seq2seq model, a tool which performs a joint modeling both for code structure and code semantics of the source code changes in the encoder. Then they applied copy mechanism (See et al., 2017) to mitigate the OOV problems in the decoder. Huang et al. (2017) proposed a method that reuse the existing comments in historical version to generate commit description. Jiang et al. (2017) built a seq2seq model with attention network and adopted this model to generate commit comments from `diffs`. Nie et al. (2020) constructed a contextualized code representation learning method which can utilize contextual information to represent code commit sequences.

Researchers also have proposed some approaches for automatic release notes generation (Abebe et al., 2016; Moreno et al., 2014, 2016). Moreno et al. (2014, 2016) constructed the first tool to generate release notes, named ARENA. ARENA first needs to summary each commit in a release and generates release note by using manually defined templates to combine these summaries according to their relevant information in the issue tracker. Abebe et al. (2016) manually analyzed multiple release notes from different software system and identified six types of information in release notes. Then they used machine learning algorithms to recommend the issues that should be contained in the release notes.

As for PRs, researchers mainly focus on how to understand PRs (Rahman and Roy, 2014; Tsay et al., 2014) and how the pull-based development work (Gousios et al., 2014, 2016), such as priority prediction of PRs (Van Der Veen et al., 2015), reviewers recommendation for PRs (Yu et al., 2014), and popularity analysis of the pull-based development model (Gousios et al., 2015). Motivated by these tasks, Liu et al. (2019b) were the first to propose the automatic PR description task in order to facilitate downstream tasks. Then they integrated pointer network (See et al., 2017) and built an attentional seq2seq model. Besides, they directly utilized the ROUGE metric to train their model by introducing SCST (Rennie et al., 2017).

Differences: Our approach is also proposed to generate the descriptions for pull requests, but it is different from the above-mentioned approaches. We mainly utilize hybrid attention network to implement this task, as a result, it shows a better performance in effectiveness and efficiency.

7.2. Summarizing other software artifacts

Except for software changes, summarizations of the source code (Hu et al., 2018; Iyer et al., 2016; Yu et al., 2020) and app reviews (Di Sorbo et al., 2017; Gao et al., 2018) are another two research hot-pots. For the source code, recent researches focus on deep learning-based models. For example, Iyer et al. (2016) proposed CODE-NN, the first model which uses an attentional seq2seq model. At the basic of CODE-NN, Hu et al. (2018) added the structure information of the source code to the attentional seq2seq model by introducing structure-base traversal method to traverse the abstract syntax tree. Yu et al. (2020) introduced graph attention network (Veličković et al., 2018) and utilized class-level contextual information to generate valid code comments. In contrast to traditional seq2seq model building by the recurrent neural network or its variants, we only use HAN to construct our seq2seq model.

As for app reviews, Di Sorbo et al. (2016) built SURF, a tool that can capture user reviews that are useful for developers via a conceptual model and generate the agenda of recommended software changes by summarizing these user reviews. Gao et al. (2018) constructed INFAR, which summaries app reviews from multiple perspectives such as salient topics and abnormal topics. **Differences**: Different from the above-mentioned studies, our approach is to automatically generate the descriptions for pull requests so that developers can easily understand what changes other developers made and the corresponding reasons. In addition, we utilize hybrid attention network to implement this goal, which shows the better performance.

8. Conclusion

In this paper, we are committed to improving the performance of PR description generation models, including effectiveness and efficiency. Thus aiming at these two aspect, we propose a novel model named PRHAN. Before and after the training phase, we use BPE to reduce the size of vocabulary and address the OOV problem. In the training phase, because PRHAN is solely based on HAN, it has faster execution efficiency and higher prediction accuracy. Besides, we utilized label smoothing to build a revised cross-entropy loss function, which can reduce the sensitivity of PRHAN and enforce its generalization ability. Experimental results on the open source dataset illustrate that PRHAN can effectively generate descriptions for PRs.

Our future works focus on two aspects. On the one hand, we will try to build a large-scale dataset and use it to develop a useful tool that can effectively generate descriptions for PRs. On the other hand, we plan to apply each component in PRHAN to other software engineering tasks such as source code summarization and bug report summarization.

CRediT authorship contribution statement

Sen Fang: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Visualization. **Tao Zhang:** Supervision, Resources, Writing - review & editing, Project administration, Funding acquisition. **You-Shuai Tan:** Writing - review & editing. **Zhou Xu:** Writing - review & editing. **Zhi-Xin Yuan:** Evaluation, Validation. **Ling-Ze Meng:** Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Science and Technology Development Fund of Macau under Grant 0047/2020/A1, in part by the China Postdoctoral Science Foundation under Grants 2017M621247 and 2020M673137, and in part by the Natural Science Foundation of Heilongjiang Province, China under Grant LH2019F008.

References

Abebe, S.L., Ali, N., Hassan, A.E., 2016. An empirical study of software release notes. Empir. Softw. Eng. (EMSE) 21, 1107–1142.

Ahmad, W.U., Chakraborty, S., Ray, B., Chang, K.-W., 2020. A transformer-based approach for source code summarization. arXiv:2005.00653.

Bahdanau, D., Cho, K., Bengio, Y., 2016. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473.

Bauer, L., Wang, Y., Bansal, M., 2018. commonsense for generative multi-hop question answering tasks. In: Proceedings of the 2018 Conference on Empirical Methods in Natural, EMNLP, pp. 4220–4230.

Conneau, A., Schwenk, H., Barrault, L., LeCun, Y., 2017. very deep convolutional networks for text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL, pp. 1107–1116.

Cortés-Coy, L.F., Linares-Vásquez, M., Aponte, J., Poshyvanyk, D., 2014. On automatically generating commit messages via summarization of source code changes. In: Proceedings of the 14th IEEE International Working Conference on Source Code Analysis and Manipulation, pp. 275–284.

Devlin, J., Chang, M., Lee, K., Toutanova, K., 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies NAACL-HLT, pp. 4171–4186.

Di Sorbo, A., Panichella, S., Alexandru, C.V., Shimagaki, J., Visaggio, C.A., Canfora, G., Gall, H.C., What would users change in my app? summarizing app reviews for recommending software changes. In: Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE, pp. 499–510.

Di Sorbo, A., Panichella, S., Alexandru, C.V., Visaggio, C.A., Canfora, G., 2017. SURF: summarizer of user reviews feedback. In: Proceedings of the IEEE/ACM 39th International Conference on Software Engineering Companion. IEEE, pp. 55–58.

Dong, L., Xu, S., Xu, B., 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In: Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5884–5888.

Fang, S., Tan, Y.-S., Zhang, T., Liu, Y., 2021. Self-attention networks for code search. Inf. Softw. Technol. 134, 106542.

Fernandes, P., Allamanis, M., Brockschmidt, M., 2021. Structured neural summarization. arXiv:1811.01824.

Gage, P., 1994. A new algorithm for data compression. C Users J. 12, 23–38.

Gao, Y., Wang, W., Herold, C., Yang, Z., Ney, H., 2020. Towards a better understanding of label smoothing in neural machine translation. In: Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, AACL/IJCNLP, pp. 212–223.

Gao, C., Zeng, J., Lo, D., Lin, C.-Y., Lyu, M.R., King, I., 2018. INFAR: Insight extraction from app reviews. In: Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE, pp. 904–907.

Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N., 2017. Convolutional sequence to sequence learning. In: Proceedings of the 34th International Conference on Machine Learning, ICML, pp. 1243–1252.

Gehrmann, S., Deng, Y., Rush, A.M., 2018. Bottom-up abstractive summarization. arXiv:1808.10792.

Gousios, G., Pinzger, M., Deursen, A.v., 2014. An exploratory study of the pull-based software development model. In: Proceedings of the 36th International Conference on Software Engineering, ICSE, pp. 345–355.

Gousios, G., Storey, M.-A., Bacchelli, A., 2016. Work practices and challenges in pull-based development: the contributor's perspective. In: Proceedings of the IEEE/ACM 38th International Conference on Software Engineering, ICSE, pp. 285–296.

Gousios, G., Zaidman, A., Storey, M.-A., Van Deursen, A., 2015. Work practices and challenges in pull-based development: the integrator's perspective. In: Proceedings of the IEEE/ACM 37th IEEE International Conference on Software Engineering, ICSE, pp. 358–368.

Gu, J., Lu, Z., Li, H., Li, V.O.K., 2016. Incorporating copying mechanism in sequence-to-sequence learning. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 1631–1640.

Gu, X., Zhang, H., Kim, S., 2018. Deep code search. In: Proceedings of the IEEE/ACM 40th International Conference on Software Engineering, ICSE. IEEE, pp. 933–944.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Hu, X., Li, G., Xia, X., Lo, D., Jin, Z., 2018. Deep code comment generation. In: Proceedings of the IEEE/ACM 26th International Conference on Program Comprehension, ICPC, pp. 200–20010.

Huang, Y., Zheng, Q., Chen, X., Xiong, Y., Liu, Z., Luo, X., 2017. Mining version control system for automatically generating commit comment. In: Proceedings of the 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 414–423.

Iyer, S., Konstas, I., Cheung, A., Zettlemoyer, L., 2016. Summarizing source code using a neural attention model. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 2073–2083.

Jiang, S., Armaly, A., McMillan, C., 2017. Automatically generating commit messages from diffs using neural machine translation. In: Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering. ASE, pp. 135–146.

Kalchbrenner, N., Blunsom, P., 2013. Recurrent continuous translation models. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP, pp. 1700–1709.

Karampatsis, R.-M., Babii, H., Robbes, R., Sutton, C., Janes, A., 2020. Big code!=big vocabulary: Open-vocabulary models for source code. In: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering, ICSE, pp. 1073–1085.

Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization. arXiv:1412.6980.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R., 2020. ALBERT: A lite BERT for self-supervised learning of language representations. arXiv:1909.11942.

Lebanoff, L., Song, K., Dernoncourt, F., Kim, D.S., Kim, S., Chang, W., Liu, F., 2019. Scoring sentence singletons and pairs for abstractive summarization. arXiv:1906.00077.

LeClair, A., Jiang, S., McMillan, C., 2019. A neural model for generating natural language summaries of program subroutines. In: Proceedings of the IEEE/ACM 41st International Conference on Software Engineering, ICSE, pp. 795–806.

LeClair, A., McMillan, C., 2019. Recommendations for datasets for source code summarization. arXiv:1904.02660.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H., Kang, J., 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics 36 (4), 1234–1240.

Li, P., Lam, W., Bing, L., Wang, Z., 2017. Deep recurrent generative decoder for abstractive text summarization. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2091–2100.

Li, H., Zheng, Y., Ren, P., 2019. Dual-channel attention model for text sentiment analysis. Int. J. Perform. Eng. 15 (3), 834–841.

Lin, C.-Y., 2004. Rouge: A package for automatic evaluation of summaries. In: Proceedings of the Text Summarization Branches Out, pp. 74–81.

Liu, S., Chen, Y., Xie, X., Siow, J., Liu, Y., 2021. Retrieval-augmented generation for code summarization via hybrid GNN. arXiv:2006.05405.

Liu, Y., Lapata, M., Text summarization with pretrained encoders. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, pp. 3728–3738.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019a. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692.

Liu, Z., Xia, X., Treude, C., Lo, D., Li, S., 2019b. Automatic generation of pull request descriptions. In: Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering, ASE, pp. 176–188.

Loyola, P., Marrese-Taylor, E., Matsuo, Y., 2017. A neural architecture for generating natural language descriptions from source code changes. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 287–292.

Luong, T., Pham, H., Manning, C.D., 2015. Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP, pp. 1412–1421.

Lutellier, T., Pham, H.V., Pang, L., Li, Y., Wei, M., Tan, L., 2020. CoCoNuT: combining context-aware neural translation models using ensemble for program repair. In: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 101–114.

Moreno, L., Bavota, G., Di Penta, M., Oliveto, R., Marcus, A., Canfora, G., Automatic generation of release notes, Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE, pp. 484–495.

Moreno, L., Bavota, G., Di Penta, M., Oliveto, R., Marcus, A., Canfora, G., 2016. ARENA: an approach for the automated generation of release notes. IEEE Trans. Softw. Eng. (TSE) 43, 106–127.

Müller, R., Kornblith, S., Hinton, G.E., 2019. When does label smoothing help? In: Proceedings of the 33rd Conference on Neural Information Processing Systems, NIPS, pp. 4694–4703.

Nie, L.Y., Gao, C., Zhong, Z., Lam, W., Liu, Y., Xu, Z., 2020. Contextualized code representation learning for commit message generation. arXiv:2007.06934.

Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., 2002. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318.

Rahman, M.M., Roy, C.K., 2014. An insight into the pull requests of github. In: Proceedings of the 11th Working Conference on Mining Software Repositories, MSR, pp. 364–367.

Rennie, S.J., Marcheret, E., Mroueh, Y., Ross, J., Goel, V., 2017. Self-critical sequence training for image captioning. In: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 7008–7024.

See, A., Liu, P.J., Manning, C.D., 2017. Get to the point: Summarization with pointer-generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 1073–1083.

Sennrich, R., Haddow, B., Birch, A., 2016. Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 1715–1725.

Shaw, P., Uszkoreit, J., Vaswani, A., 2018. Self-attention with relative position representations. arXiv:1803.02155.

Stapleton, S., Gambhir, Y., LeClair, A., Eberhart, Z., Weimer, W., Leach, K., Huang, Y., 2020. A Human study of comprehension and code summarization. In: Proceedings of the 28th International Conference on Program Comprehension, pp. 2–13.

Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks. In: Proceedings of the 28th Conference on Neural Information Processing Systems, NIPS, pp. 3104–3112.

Tsay, J., Dabbish, L., Herbsleb, J., 2014. Influence of social and technical factors for evaluating contribution in GitHub. In: Proceedings of the 36th International Conference on Software Engineering, ICSE, pp. 356–366.

Van Der Veen, E., Gousios, G., Zaidman, A., 2015. Automatically prioritizing pull requests. In: Proceedings of the IEEE/ACM 12th Working Conference on Mining Software Repositories, MSR, pp. 357–361.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. In: Proceedings of the 31st Conference on Neural Information Processing Systems, NIPS, pp. 5998–6008.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y., 2018. Graph attention networks. arXiv:1710.10903.

Vinyals, O., Fortunato, M., Jaitly, N., 2015. Pointer networks. In: Proceedings of the 29th Conference on Neural Information Processing Systems, NIPS, pp. 2692–2700.

Wang, Y., Chen, T., Xu, H., Ding, S., Lv, H., Shao, Y., Peng, N., Xie, L., Watanabe, S., Khudanpur, S., 2019. Espresso: A fast end-to-end neural speech recognition toolkit. In: Proceedings of the 2019 IEEE Automatic Speech Recognition and Understanding Workshop, pp. 136–143.

Wei, B., Li, Y., Li, G., Xia, X., Jin, Z., 2020. Retrieve and refine: exemplar-based neural comment generation. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, ASE, pp. 349–360.

Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J., 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144.

Xu, M., Wong, D.F., Yang, B., Zhang, Y., Chao, L.S., 2019a. Leveraging local and global patterns for self-attention networks. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 3069–3075.

Xu, S., Yao, Y., Xu, F., Gu, T., Tong, H., Lu, J., 2019b. Commit message generation for source code changes. In: Proceedings of the 2019 28th International Joint Conference on Artificial Intelligence, IJCAI, pp. 3975–3981.

Yang, B., Tu, Z., Wong, D.F., Meng, F., Chao, L.S., Zhang, T., 2018. Modeling localness for self-attention networks. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP, pp. 4449–4458.

Yu, A.W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., Le, Q.V., 2018. QANet: Combining local convolution with global self-attention for reading comprehension. arXiv:1804.09541.

Yu, X., Huang, Q., Wang, Z., Feng, Y., Zhao, D., 2020. Towards context-aware code comment generation. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pp. 3938–3947.

Yu, Y., Wang, H., Filkov, V., Devanbu, P., Vasilescu, B., 2015. Wait for it: Determinants of pull request evaluation latency on github. In: Proceedings of the IEEE/ACM 12th Working Conference on Mining Software Repositories, MSR, pp. 367–371.

Yu, Y., Wang, H., Yin, G., Ling, C.X., 2014. Reviewer recommender of pull-requests in github. In: Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution, pp. 609–612.

Zhang, T., Chen, J., Jiang, H., Luo, X., Xia, X., 2017. Bug report enrichment with application of automated fixer recommendation. In: Proceedings of the IEEE/ACM 25th International Conference on Program Comprehension, ICPC, pp. 230–240.

Zhang, H., Goodfellow, I., Metaxas, D., Odena, A., 2019. Self-attention generative adversarial networks. In: Proceedings of the 36th International Conference on Machine Learning, ICML, pp. 7354–7363.

Zhao, C., Wang, J., 2019. Service recommendation model based on rating matrix and context-embedded lstm. Int. J. Perform. Eng. 15, 2432–2431.

Sen Fang is a Ph.D. student in the Faculty of Information Technology, Macau University of Science and Technology (MUST), under the supervision of Prof. Tao Zhang. Before joining MUST, he got M.Sc. in Electronics and Communication Engineering from Central China Normal University in 2020. His research interests lie in software engineering and NLP, particularly using NLP technologies to build effective models for representing the source code.

Tao Zhang received the BS degree in automation, the M.Eng. degree in software engineering from Northeastern University, China, and the Ph.D. degree in computer science from the University of Seoul, South Korea. After that, he spent one year with the Hong Kong Polytechnic University as a postdoctoral research fellow. Currently, he is an associate professor with the Faculty of Information Technology, Macau University of Science and Technology (MUST). Before joining MUST, he was the faculty member of Harbin Engineering University and Nanjing University of Posts and Telecommunications, China. He published more than 60 high-quality papers at renowned software engineering and security journals and conferences such as the IEEE Transactions on Software Engineering, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing, IEEE Software, ICSE, etc. His current research interests include AI for software engineering and mobile software security. He is a senior member of IEEE and ACM.

Youshuai Tan is a postgraduate student at the Faculty of Information Technology, Macau University of Science and Technology (MUST), under the supervision of Prof. Tao Zhang. He obtained his B.Eng. from Harbin Engineering University in 2021. His works have been published in JSS and TR. His current research interests include software engineering and natural language processing.

Zhou Xu is an assistant professor in the School of Big Data and Software Engineering at Chongqing University, China. He received two Ph.D. degrees from Wuhan University (Wuhan, China) and The Hong Kong Polytechnic University (Hong Kong, China) in 2019 and 2021, respectively. His research interests include software defect prediction, empirical software engineering, feature engineering, and data mining.

Zhixin Yuan is a software engineer in the United Imaging Healthcare Corporation (UIH). Before joining UIH, he got M.Sc. in Computer Science and Information Engineering from Hubei University (Wuhan, China) in 2021. His research interests lie in satellite navigation signal processing and NLP.

Lingze Meng is a postgraduate student in the Faculty of Information Technology, Macau University of Science and Technology (MUST), under the supervision of Prof. Tao Zhang. His research interests include NLP, software bug localization, and software bug prediction.