2018 25th Asia-Pacific Software Engineering Conference (APSEC)

Improving Bug Localization with Character-level Convolutional Neural Network and Recurrent Neural Network

Yan Xiao, Jacky Keung
Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
Email: yanxiao6-c@my.cityu.edu.hk, Jacky.Keung@cityu.edu.hk

***Abstract*—*Background:* Automated bug localization in large amounts of source files for bug reports is a crucial task in software engineering. However, the different representations of bug reports and source files limited the accuracy of the existing bug localization techniques. *Aims:* We propose a novel deep learning-based model to improve the accuracy of bug localization for bug reports by expressing them in character and analyzing them with a language model. *Method:* The proposed model is composed of two main parts: character-level convolutional neural network (CNN) and recurrent neural network (RNN) language model. Both bug reports and source files are expressed in a character level and then input into a CNN, whose output is given to an RNN encoder-decoder architecture. *Results:* The results of preliminary experiments show that the proposed model achieves comparable or even higher accuracy than the existing machine translation-based bug localization technique. *Conclusion:* The proposed model is capable of automatically localizing buggy files for bug reports and achieves better accuracy by analyzing them in character level where both bug reports and source code can be expressed.**

***Index Terms*—bug localization, convolutional neural network, recurrent neural network, deep learning**

I. Introduction and Motivation

Automatically localizing buggy files for bug reports remains a significant task in software project teams, especially those involving hundreds of thousands of source files. It is painstaking for developers to search all source files for bug-fixing. The automated bug localization techniques are thus proposed to rank the source files and recommend the top relevant files to developers. However, bug reports are written in natural languages while source files are written in code tokens. The different expressions between them have been empirically demonstrated to be responsible for the low accuracy of the existing bug localization techniques [1], [2], [3], [4].

Ye et al. [2] tried to bridge the lexical gap by adding the semantic similarity between bug reports and source files into their previous proposed learning-to-rank model [1]. Xiao et al. [3] transformed bug reports and source files into word vectors using word embedding techniques to preserve the semantics, and extracted features from word vectors using enhanced CNN. To distinguish bug reports and source files, Xiao et al. [4] proposed BugTranslator, a machine translation-based bug localization technique. However, all the existing techniques regard both bug reports and source files as natural languages.

The code tokens in source files are similar to English words in natural languages while the difference is obvious. Some code tokens, especially the class or method names, are not actual words that are commonly used in natural languages. Although they are very important in bug localization, most existing studies regarded them as unknown words [2], [3], [4]. However, both bug reports and source files are composed of characters. They share same expressions in character level. Therefore, this paper proposes a bug localization technique based on a language model in character-level. The proposed model first obtains the character embeddings of the preprocessed bug reports and source files. Two CNNs with multiple filters are then applied to extract features respectively from the vectors of bug reports and source files, whose outputs are fed into the subsequent RNN encoder-decoder architecture. The experiments on three open-source Java projects show the feasibility and effectiveness of the proposed model.

II. The Proposed Model

This section describes the proposed model whose overview is illustrated in Figure 1.

A. Data Preprocessing

We first combine summary and description in bug reports to be a new document. Revised term frequency-user focused inverse document frequency (TF-IDuF) is then applied to filter some common words in the new bug reports for the purpose of redundancy reduction [3]. We also extract two types of Abstract Syntax Tree nodes from each source file as [4].

B. Character-level CNN

After preprocessing bug reports and source code, convolutional operations are applied in each word of them respectively. Each character in a word is transformed into a k-dimensional character embeddings, which are then convolved by multiple filters with different sizes ($2 \times k$ and $3 \times k$ in Figure 1). The subsequent max-pooling layer is used to conclude the features, whose outputs are given to encoder and decoder respectively.

C. RNN Encoder-Decoder

1) Encoder: The output features extracted from a word in bug reports by the character-level CNN are given to one long short-term memory (LSTM) cell. For example, the feature

978-1-7281-1970-0/18/$31.00 ©2018 IEEE
DOI 10.1109/APSEC.2018.00097

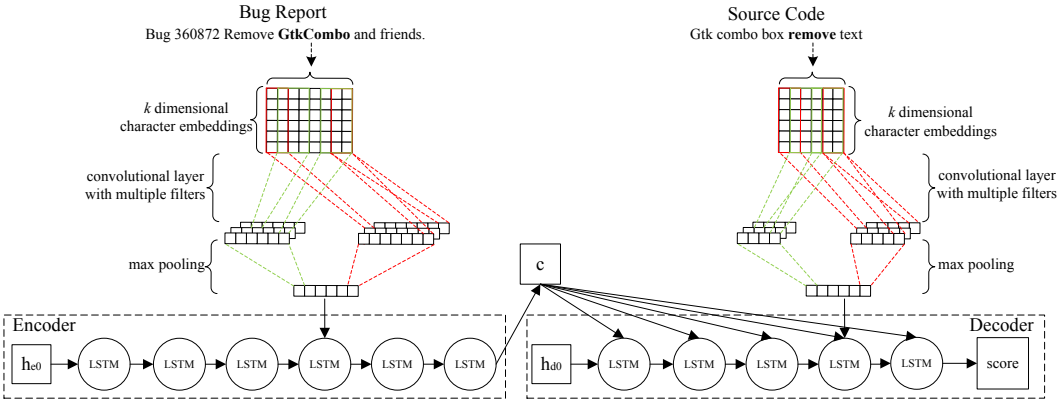

Fig. 1. The overview of the proposed model.

vectors of the fourth word in a bug report are fed into the fourth LSTM in the encoder as shown in Figure 1. The context vector c is the final state of the encoder, which is the conclusion of the features of bug reports that will be one part of input in the decoder.

2) Decoder: Similar to the encoder, the features extracted from each word in source code by the character-level CNN are one part of the input to each LSTM cell. Besides, the context vector is concatenated with output features of each word to be fed into each LSTM.

III. The Preliminary Results

In order to validate the feasibility and effectiveness of the proposed model, we conduct several preliminary experiments on the before-fixed version of three open-source Java projects [1] similar to [4]. 3656, 2632, 2817 bug reports respectively for Project Eclipse UI, JDT, SWT are used. Mean average precision (MAP) and mean reciprocal rank (MRR) are used to evaluate the performance of the proposed model and the competitor BugTranslator [4].

TABLE I
Results of Two Models.

Project	Metrics	BugTranslator	Proposed model
Eclipse UI	MAP	0.36	0.35
	MRR	0.42	0.40
JDT	MAP	0.34	0.35
	MRR	0.41	0.42
SWT	MAP	0.34	0.37
	MRR	0.40	0.42

The preliminary results are shown in Table I. The MAP and MRR values of the proposed model are better than BugTranslator in Project JDT and SWT. The performance of BugTranslator is limited since it analyzes bug reports and source files in word level where many out-of-vocabulary words exist. But BugTranslator gives more emphases on the related words in buggy files with those in bug reports using an attention mechanism. It is much important when there are large amounts of source files. In Project Eclipse UI, the number of source files is 6228 that is about five times the number in Project SWT. The performance of BugTranslator is thus better than our proposed model in Project Eclipse UI.

[1]https://github.com/yanxiao6/BugLocalization-dataset

IV. Conclusions and Future Work

In this paper, bug reports and source files are analyzed in character-level instead of word-level to suppress the effect of different expressions on the accuracy of bug localization. The number of unknown words is also reduced. The proposed model applies character-level CNN to extract features from bug reports and source files, whose output is fed into the subsequent RNN encoder-decoder. The preliminary results indicate the feasibility and effectiveness of the proposed model.

We intend to enhance the proposed model with attention mechanisms and fine-tune the proposed model. In the future, we will conduct experiments on more projects to obtain the general performance of the proposed model.

V. Acknowledgement

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No. 11208017) and the research funds of City University of Hong Kong (No. 9678149 and 7005028), and the Research Support Fund by Intel.

References

[1] X. Ye, R. Bunescu, and C. Liu, "Learning to rank relevant files for bug reports using domain knowledge," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 689–699.

[2] X. Ye, H. Shen, X. Ma, R. Bunescu, and C. Liu, "From word embeddings to document similarities for improved information retrieval in software engineering," in *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2016, pp. 404–415.

[3] Y. Xiao, J. Keung, Q. Mi, and K. E. Bennin, "Improving bug localization with an enhanced convolutional neural network," in *Asia-Pacific Software Engineering Conference (APSEC), 2017 24th*. IEEE, 2017, pp. 338–347.

[4] Y. Xiao, J. Keung, K. E. Bennin, and Q. Mi, "Machine translation-based bug localization technique for bridging lexical gap," *Information and Software Technology*, 2018.