Using Document Embedding Techniques for Similar Bug Reports Recommendation

Ming Chen, Dongyang Hu, Tao Wang, Jun Long,Gang Yin, Yue Yu, Yang Zhang
Science and Technology on Parallel and Distributed Processing Laboratory
National University of Defense Technology, Changsha, China
{chenmingI7c, hudongyangl7, taowang2005, junlong, jack_nudt, yueyu, yangzhang15}@ nudt.edu.cn

Abstract– In the software development, it is common to find that several bug reports are related to many common code files, Le., similar bugs. Similar bug recommendation is a meaningful task that could assist developers in bug triaging and fixing. If developers can focus on fewer code files, they can fix similar bugs in less time with higher quality. Therefore, building a sitnilar bug recommendation system is a meaningful task that can improve development efficiency. As the state of the art, Yang et al.'s work presented an approach that combines TF-IDF method with Word Embedding model and achieved a good result. To further improve the performance of their approach, in this paper, **we** propose a novel approach using Document Embedding model. In our preliminary evaluation, **we** conduct the experiment on the datasets of Jdt, Birt and Eclipse platform. And the results show that our approach outperforms Yang et al.'s, with 2.67-9.50% of improvement.

Keywords-component; Similar Bugs; Document Embedding; Bug recommendation systems;

I. INTRODUCTION

In the software development process, bug report is an important type of knowledge and many software projects may receive a lot of new bug reports every day. Thus, it is common to find that several bug reports are related to many common code files, i.e., similar bugs. One manifestation of this is that many open bugs in one project may get linked to related bugs [1]. Developers can often fix similar bugs with a shorter time and a higher quality since they can focus on fewer code files [2]. Thus, similar bug recommendation is a meaningful task that can assist developers in bug triaging and fixing, thereby improving development efficiency.

Rocha et al. proposed the first similar bug recommendation system named NextBug [3]. Recently, Yang et al. proposed an enhanced approach to recommend similar bugs by combining traditional information retrieval technique (TF-IDF) with Word Embedding model [2]. However, we find that there is still space for improvement in Yang et ai' s approach. To further improve the performance of their approach, we propose a novel approach using Document Embedding model in this paper, which considers the latent relationship between two bug reports at the document-level.

To initially evaluate the performance of our approach, we conduct a preliminary experiment on the dataset of Jdt, Birt and Eclipse platform1 (Jdt: 10,494; Birt: 8,913; Eclipse: 13,090), and compare our approach with Yang et al.'s.

In the evaluation, we use the same evaluation metrics as Yang et al.s, *i.e.; Recall-Rate@k, Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR)*. These metrics are commonly used in evaluating recommendation systems in software engineering tasks.

The experimental results show that our approach outperforms Yang et al.'s approach in terms of all metrics, with 2.67-9.50% of improvement.

The main contributions of this paper are:

1) We propose a novel approach using Document Embedding model to recommend similar bugs. This paper is an enhanced version of our previous work[12].

2) We compare our approach with Yang et al.'s approach on Jdt, Birt and Eclipse platform. The results show that our approach achieves a significant improvement over Yang et al.'s approach.

This paper is organized as follows. Section II describes related works. Section III introduces our approach and Section IV presents our preliminary study results. Section V concludes this paper.

II. RELATED W ORKS

In this section, we first introduce the related works about similar bug recommendation, then we present other works about bug report management. At last, we highlight some software engineering studies that leverage Word Embedding.

A. Similar Bug

In bug tracking systems, similar bugs refer to bugs that need to handle many common source code files. Similar bugs are related to duplicate bugs, which are widely studied in the field of software engineering [4][14][15]. However, the condition for two bugs being similar is much looser than that for duplicate bugs, which means that there are more similar bugs than duplicate bugs, given a query bug.

Recently, some researchers began to investigate the impact of related bugs [1] and bug related resources (e.g., StackOverflow posts [5]) on the triaging and fixing of bug reports. Researchers have developed different methods that extracting textual information from bug reports and sources files, and recommend similar bugs by ranking the textual similarity for all bugs.

B. Bug report management with machine learning

There are a number of studies on bug localization by leveraging machine learning algorithms. Kim et al. [6] applied Naive Bayes using previously fixed files as classification labels and use the trained model to assign source files to each report. An Ngoc Lam et al. [7] proposed a new approach that uses neural network in combination with rVSM, an information retrieval technique. First, they use rVSM to collect the feature on the textual similarity between bug reports and sources files. Then they use DNN to learn to relate the terms in bug reports to potentially different code tokens and terms in source files and documentation if they appear frequently enough in the pairs of reports and buggy files. Ye et al. [8] leverage features from source files, API description, bug fixing and change history to compute ranking score of each source file for a given bug report.

C. Bug recommendation with Word Embedding

In the natural language processing (NLP) area, a large number of studies have been conducted to characterize words' context and capture latent semantics by using neural-network-based embedding vectors, (e.g., Word Embedding [9] and Document Embedding [10]). Recently, some researchers began to leverage these neural embedding techniques to solve software engineering tasks and obtained good performances [2], [11]. Chen et al. [13] proposed a new method that incorporates relational and categorical knowledge into Word Embedding to recommend analogical libraries in Q&A discussions. Yang et al. [2] presented a novel approach by combining traditional information retrieval technique and Word Embedding to recommend similar bugs.

III. OUR APPROACH

First, we introduce some background knowledge related to our study and present an overall framework of our approach.

A. Word Embedding

Word Embedding models use deep neural networks in order to represent word with a word vector and focus more on the relationship of words. [9] CBOW model and skip-gram model are two main Word Embedding models applied to many information retrieval tasks. The objective of the CBOW model is to predict the current word based on its context while the skip-gram model is aimed at predicting the current surrounding words for the given word.

After training, each word is mapped to a low-dimensional vector. The results show that words with similar meaning are much closer with each other than those with different meaning. And the distance between those vectors also carry meaning which provides us a possibility to utilize linear matrix to transform words and phrases between languages. Due to these properties, it is practical for us to apply this model to recommend similar bugs according to their text information and make a better understanding of bug itself.

B. Document Embedding

Document Embedding model was proposed by Quoc Le and Tomas et aL [10]. It can transform variable length pieces of texts into continuous distributed vector. Document Embedding is inspired by the Word Embedding model. Similar to Word Embedding model, there are also two popular Document Embedding models, i.e., DM model and DBOW model. Every column in document matrix D represents a unique document vector and every column in matrix *W* represents a unique word vector as welL The document token is used to remember what is missing from the current context. By sampling from a sliding window over the document we can obtain the fixed-length contexts. Noticed that the document vector is shared across all contexts which are generated from the same document. However, the word vector matrix *W* is shared across documents.

The algorithm of DM model is divided into two steps:
1) Training the document vectors and word vectors by using the stochastic gradient descent.
2) Getting document vectors D for documents that never seen before.

Contrary to the DM model, DBOW model just predict words sampled from document. One of the most important advantages of document vectors is that they perform pretty well for tasks that do not have enough labeled data.

C. *TF-IDF*

TF-IDF (Term Frequency-Inverse Document Frequency) is an importance information retrieval technique which has been widely used in many areas. The purpose of TF-IDF is to reflect the importance of a word to a document in a collection or corpus. For a given term t and a document d, the definitions of TF and IDF are as follows [2]:

$$TF(I, d) = \frac{Number\ of\ limes\ I\ appears\ in\ d}{Number\ of\ lerms\ in\ d} \qquad (1)$$

$$JDF(t) = \log \frac{Total\ number\ of\ documents}{Number\ of\ documents\ that\ contains\ t + 1} \qquad (2)$$

According to the formulation above we can compute TF-IDF as:

$$TF - IDF(t,d) = TF(t,d) \times IDF(t) \qquad (3)$$

Given a document d, we can convert it into a TF-IDF vector with the formulation above. And we can compute the cosine similarity of two document with their TF-IDF vector.

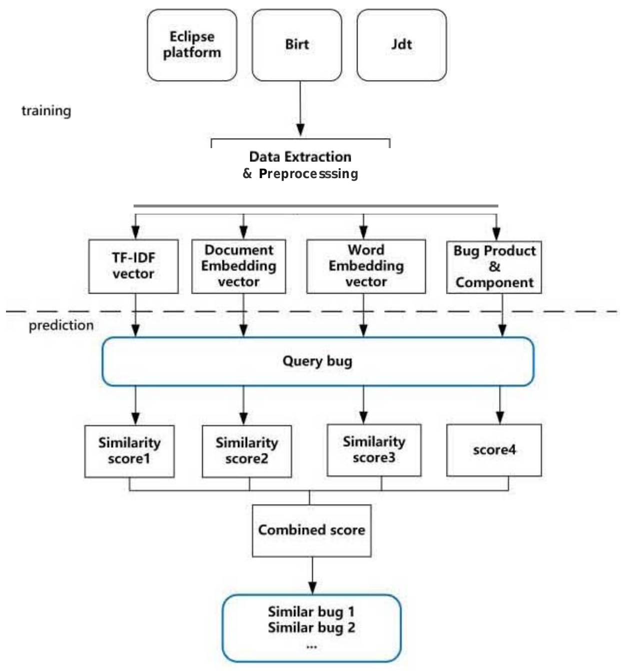

Figure 1. Overview of our approach

Figure 1 presents the framework of our approach, which is mainly based on the state of the art approach proposed by Yang et al. [2]. The framework mainly contains two stages i.e., training stage and prediction stage. In training stage, this approach first extracts textual information (i.e., bug title and description) from original dataset and creates bug documents. Next, our approach begin to train the TF-IDF model, Document Embedding model and Word Embedding model based on the bug documents (i.e., bug title and description). In prediction stage, given a query bug, this approach first converts it into TF-IDF vector, Document Embedding vector and Word Embedding vector. Next, our approach calculates the cosine similarity of the query bug with each of pending bugs based on TF-IDF vectors, Document Embedding vectors, and Word **Embedding vectors, to generate similarity scores *Score],* *Score2,and Score3.* Based on the product and component** information for the query bug and each of pending bug, we **generate *Score4.* Finally, our approach calculates a final score,** i.e., *Score = (Score] +Score2+Score3)*Score4.* The higher the **score, the more similar the corresponding pending bug is with** the query bug. After sorting the pending bugs based on the **final scores, we recommend the most similar k bugs respect to** a given bug (i.e., k-l, 5, 10).

D. *Evaluation Metrics*

Given a query bug *q,* S(*q)* and *R(q)* denotes its ground truth similar bug set and the top-k recommendation set generated by **a recommendation system respectively. We define the *Recall-*** *Rateesk, Mean Average Precision (MAP)* and *Mean Reciprocal Rank (MRR)* based on these notations as follows [2].

1) Reeall-Rate@k. The definition of *Recall-Role@k* for a query bug *q* is as follows:

$$Recall\text{-}rate@k(q)=\begin{cases} l & if\ S(q)n\ R(q)\neq\varnothing \\ 0 & Otherwise \end{cases} \quad (4)$$

According to the formulation above, given a set of query bugs, we compute the proportion of useful top-k recommendations by averaging the *Recall-Roles@k* of all query bugs to get an overall *Recall-Role@k.*

2) MAP. MAP (Mean Average Precision) is defined as the mean of the Average Precision (AvgP) values obtained for all the evaluation queries. The AvgP of a given query *q* IS calculated as follows:

$$AvgP(q)=\sum_{n=1}^{|S(q)|}\frac{Prec@k(q)}{|Seq)|} \quad (5)$$

In the above formula, *Prec@k* denotes the ratio of ground truth similar bugs of the query bug *q* in the top K **recommendation:**

$$Prec@k(q)=\frac{\#\ of\ ground\ truth\ similar\ bugs\ in\ top-k}{n} \quad (6)$$

3) MRR. MRR (Mean Reciprocal Rank) is defined as the mean of the Reciprocal Rank (RR) values obtained for all the **evaluation queries, where RR of a single query *q* is the multiplicative inverse of the rank of the first correct answer** *first.;* (i.e., first ground truth similar bug m the **recommendation list):**

$$RR(q)=\frac{1}{first_q} \quad (7)$$

IV PRELIMINARY EVALUATION

In this section, we conducted several experiments aiming to compare the effectiveness of our approach with Yang et al.'s on Jdt, Birt and Eclipse Platform.

A. *Experimental Setting*

In our preliminary evaluation, we use the data of bug reports from Jdt, Birt and Eclipse Platform. In our experiment, **given a bug *q,* we recommend similar bugs based on the** similarity scores calculated by our approach. After ranking all the pending bugs, we recommend k most similar bugs according to their similarity scores (i.e., k=1,5,10). We **compare our approach against Yang et al.'**s **and use the same metrics to evaluate the** performance **i.e., *Recall-Rate@k, Mean*** *Average Precision (MAP) and Mean Reciprocal Rank (MRR).*

B. *Experiment Results*

We evaluate our approach on three datasets, Jdt, Birt and Eclipse Platform. The experimental results are shown in Table I, Table II and Table III respectively. For Jdt dataset, our approach achieves 0.2859 of *Recall-Role@lO* and 0.2180 of *Recall-Role@5,* with a relative 9.50% and 8.62% of **improvement compared to the Yang et al.'s approach** respectively. For Birt dataset, our approach achieves 0.3154 of

Recall-Rate@lO and 0.2379 of *Recall-Rate@5,* with a relative 2.67% and 3.08% of improvement compared to the Yang et al.'s approach respectively. For Eclipse Platform dataset, our approach achieves 0.2830 of *Recall-Rate@lO* and 0.2183 of *Recall-Rate@5,* with a relative 7.89% and 8.77% of improvement compared to the Yang et al.'s approach respectively.

Among all the three datasets, our approach improves Yang et al.'s approach in terms of *Recall@5* and *Recall@lO* by 6.82% and 6.69% respectively. The values of *Recall@5, Recall@10, MAP*, and *MRR* of our approach are higher than Yang et al.'s.

TABLE I. EVALUATION RESULTS IN JDT DATASET

Project	Yang et ai's	Our approach	Improvement
Recall-Rate@1	8.83%	9.61%	8.12%
Recall-Rate@5	20.07%	21.80%	8.62%
Recall-Rate@IO	26.11%	28.59%	9.50%
MAP	6.22%	6.75%	8.52%
MRR	11.74%	12.80%	9.03%

TABLE II. EvALUATION RESULTS IN BIRT DATASET

Project	Yang et ai's	Our approach	Improvement
Recall-Rate@1	8.97%	9.54%	6.35%
Recall-Rate@5	23.08%	23.79%	3.08%
Recall-Rate@IO	30.72%	31.54%	2.67%
MAP	6.72%	7.02%	4.46%
MRR	12.65%	13.14%	3.73%

TABLE III. EVALUATION RESULTS IN ECLIPSE DATASET

Project	Yang et ai's	Our approach	Improvement
Recall-Rate@1	8.83%	9.58%	8.49%
Recall-Rate@5	20.07%	21.83%	8.77%
Recall-Rate@IO	26.23%	28.30%	7.89%
MAP	6.23%	6.75%	8.35%
MRR	11.72%	12.77%	8.96%

V. CONCLUSION AND FUTURE WORK

To assist developers in bug triaging and fixing, similar bugs recommendation is a meaningful task in software development. To validate whether our approach performs better than Yang et al.'s. We conduct our experiments on three different datasets, that is, Jdt, Birt and Eclipse platform. And we found that our approach achieves *Recall@5* scores of 0.2180, 0.2379 and 0.2183, *andRecall@10scores* of 0.2859, 0.3154 and 0.2830 on Jdt, Birt and Eclipse platform, respectively. In terms of *Recall@5* and *Recall@10,* the results show that our approach outperforms Yang et al.'s by achieving 6.82% and 6.69% improvement, respectively.

In the future, we plan to improve the performance of our approach by investigating the cases that our approach failed and applying a more sophistical machine learning model.

ACKNOWLEDGMENT

This work was supported by Laboratory of Software Engineering for Complex Systems as well as the National Grand R&D Plan (Grant No. 20l8YFBlO03903), and the National Natural Science Foundation of China (Grant NO.61502512 and 61432020).

REFERENCES

[II Y. Zhang, Y. Yu, H. Wang, B. Vasilescu, and V. Filkov, "Within-ecosystem issue linking: a large-scale study ofrails," in Software Mining. ACM, 2018, pp. 12-19.

[21 X. Yang, D. Lo, X. Xia, 1. Bao, and J. Sun, "Combining Word Embedding with information retrieval to recommend similar bug reports," in ISSRE. IEEE, 2016, pp. 127- 137.

[31 H. Rocha, M. T. Valente, H. Marques-Neto, and G. C.Murphy, "An empirical study on recommendations of similar bugs," in SANER, vol. 1. IEEE, 2016, pp. 46-56.

[41 X. Wang, 1. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in ICSE. ACM, 2008, pp. 461-470.

[51 Y. Zhang, G. Yin, T. Wang, Y. Yu, and H. Wang, "Evaluating bug severity using crowd-based knowledge: an exploratory study," in Internetware. ACM, 2015, pp. 70-73 .

[61 D. Kim, A. Zeller, Y. Tao, and S. Kim, "Where should we fix this bug?: A two-phase recommendation model," IEEE transactions on software Engineering, vol. 99, no. I, p. I, 2013.

[71 A. N. Lam, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "Combining deep learning with information retrieval to local-ize buggy files for bug reports (n)," in ASE, 2015, pp. 476-481.

[81 X. Ye, R. Bunescu, and C. Liu, "Learning to rank relevantfiles for bug reports using domain knowledge," in FS. ACM, 2014, pp. 689-699.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in NIPS, 2013, pp. 3111-3 119.

[10] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in ICML, 2014, pp. 1188-1196.

[11] B. Xu, D. Ye, Z. Xing, X. Xia, G. Chen, and S. Li, "Predicting emantically linkable knowledge in developer online forums via convolutional neural network," in ASE. ACM, 2016, pp.51-62

[12] Dongyang Hu, Ming Chen, Tao Wang, Junsheng Chang, Gang Yin, Yue Yu and Yang Zhang "Recommending Similar Bug Reports: A Novel Approach Using Document Embedding Model", in APSEC. IEEE, 2018.

[13] Chen, Chunyang, Sa Gao, and Zhenchang Xing. "Mining analogical libraries in q&a discussions--incorporating relational and categorical knowledge into word embedding." Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on. Vol. 1. IEEE, 2016.

[14] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in ICSE. IEEE, 2007, pp.499-5 10.

[15] C. Sun, D. Lo, X. Wang, J. Jiang, and S..C Khoo, "Adiscriminative model approach for accurate duplicate bug report retrieval," in ICSE. ACM,2010,pp.45-54.