International Journal of Science and Engineering Applications
Volume 12-Issue 01, 108 – 111, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1201.1035

The Impact of GloVe and Word2Vec Word-Embedding Technologies on Bug Localization with Convolutional Neural Network

Ahmed Sheikh Al-Aidaroos
Department of Information Technology
Al-ahgaff University
Hadhramout, Mukalla, Yemen

Sara Mohammed Bamzahem
Department of Information Technology
Al-ahgaff University
Hadhramout, Mukalla, Yemen

Abstract: In the field of software engineering, software quality assurance faces many challenges, including overcoming the problem of identifying errors in the source code. Finding the location of the error in the source code is a very important process, as is taking advantage of the semantic information available in the bug reports and the source code to find the similarities between them, using modern techniques called word embedding. This study aims to demonstrate how GloVe and Doc2Vec word-embedding technologies affect bug localization accuracy and performance. Therefore, this study proposes to adapt DeepLoc by using GloVe embedding techniques to process the source code instead of Word2vec and using Word2vec embedding techniques to process the bug report instead of Sent2Vec. AspectJ represents the large dataset, which contains many bug reports, while SWT's small dataset contains fewer bug reports. Experimental results show that the improved DeepLoc on SWT achieves 0.60 and 0.72 MAP and MRR, respectively. While the improved DeepLoc on AspectJ achieves 0.17 and 0.27 MAP and MRR, respectively. The results of the improved DeepLoc should be compared using two advanced models from previous studies: DeepLoc, DeepLocator.

Keywords: Bug Localization; Deep Learning; Word Embedding Techniques, GloVe Technique, Doc2Vec Technique.

1. INTRODUCTION

Software developers depend on the software life cycle when developing their software, which consists of several phases, including the testing and maintenance phases. However, after the software is used by real users, some errors or unexpected behaviors in the software's tasks are known as bugs. In this case, users write a report describing these errors, and thus the program must enter the maintenance phase.[1]. Localizing bugs requires a lot of effort and takes a lot of time on the part of software developers, so its cost is very expensive.[2]. Therefore, there is a need to facilitate the fault localization process to save effort, cost, and time. Moreover, bug reports are written in natural language, while source code files are written in programming languages. Therefore, a solution to the language incompatibility between natural languages and programming languages is necessary. Thus, several solutions have been proposed to localize the errors [2-6].

NOPL [3] is one of these solutions. By utilizing the angelic localization algorithm, NOPL takes a buggy program and test suite as input and generates a debug with a conditional expression such as if then, and else statements as output. Although NOPL effectively successfully fixed bugs related to conditional if statements in Java, it did not consider the rest of the other code statements in Java. Moreover, NOPL is limited to Java only and does not recognize other programming languages. Also, in some test cases, the inability to set the maximum execution time is caused by an angelic fix localization causing an infinite loop. Another study [4] proposed a new paradigm of information-theoretic infrared methods to support error localization tasks in software systems and aim to establish accurate semantic similarity relationships between source code and bug reports. These methods, including Mutual Information (PMI) and Uniform Google Distance (NGD), exploit coexistence patterns of code terms in a software system to reveal hidden textual semantic dimensions that other methods often fail to capture. Furthermore, the study [5] DeepLocator, a deep learning-based model, was proposed to improve error localization performance through semantic information in error reports. This is done using Word2Vec word embedding technology to handle source code and bug reports. However, the approach can be affected by derivation and the removal of stop words. Some reports are written in long terms or in an incomprehensible language, affecting the results' quality. Recently, a newer version of DeepLocator called DeepLoc was released[6], a model that makes full use of semantic information It processes bug reports and each line of source code into vectors and retains the semantics of the sentence in the vector. These vectors are then fed into CNNs to extract their hidden semantics and properties and discover the correlation between the feature vectors extracted from the bug reports and the source code. Thus, DeepLoc was using Word2Vec for source code and Sent2Vec for bug reporting. Although the study showed that using Word2Vec to process the source code is better than Sent2Vec because the source code contains keywords such as "public," "for," "void," and "int). However, the study [7] confirmed that using GloVe gives better and faster results than Word2Vec. Thus, as a first scenario, this study proposed using GloVe word embedding technology to process source code, while Word2Vec word embedding technology was used to process bug reports. In addition, another study [2] proposed a model that takes advantage of different script properties of error reports and source files as well as relationships between previously fixed error reports. Therefore, the study used one of the word embedding techniques, the global vector, for source code processing and bug reporting. However, the quality of bug reports, identifier naming conventions, and annotation methods in source files pose a threat to external validity. Also, open-source datasets are set in size, and if they are written in a language with software other than Java, this affects the quality of the results. Thus, the Doc2Vec [8] is one of the best ways that takes much less time to complete the processing process, so it is one of the best options to solve the time and speed challenge. The study also confirmed that the Doc2Vec model is much faster to build than traditional methods such as Word2Vec or Fast text word embedding techniques. Therefore, as a second scenario, this study proposed to utilize the GloVe word embedding technique to process the source code, while the Doc2Vec word embedding technique was used to process the bug reports.

2. LITERATURE REVIEW

Many studies have discussed bug localization in different approaches [2-6, 9-12]. Generally, these approaches can be categorized into 3 categories: the traditional program analysis ,machine learning & (information retrieval), and deep learning. even though the study discussed the most important achievements and challenges in the field of research, it was absent from many other contributions in the field.

One study in [3] Which uses an angelic localization algorithm to fix errors in conditional statements such as if, then and so on clauses. This, named NOPL. However, it does not take into account the rest of the other statements in Java. Moreover, NOPL is not limited to Java only it recognizes other programming languages. Also, it causes a defective patch localization.

Khatiwada et al. in [4] proposed a model of infrared informatics methods to support error localization tasks in software systems. Although the study succeeded in determining the localization of the error by arranging the files using IR techniques, it affected the error tracking in the software system to understand the cause of the error and isolate the relevant parts. While this process can be feasible when analyzing smaller systems, analyzing relatively large and complex systems can be tedious and error prone. Moreover, [9] proposed text retrieval (TR), where a search source code is indexed, which is then queried for the relevant code file for a given bug report. Although the study succeeded in determining the localization of errors through information retrieval techniques, However, TR-based methods showed poor performance when using all-text in bug reporting.

Gharibi et al. [2] proposed a text properties model is a multi-component approach to error localization that takes advantage of the unique text properties of error reports and source files and word-embedding techniques that arrange the relevant source files for each error report and then search for similarities between the source code files and the error report so that it can be reached where it has a relationship to the error report in the source code. In this way, however, the quality of error reports, identifier naming conventions, and the way comments are written in source files pose a threat of incorrect results. Furthermore, DeepLocator[5] proposed a deep learning-based model for semantics in error reports and source code. DeepLocator bridges the semantic gap by using an Abstract Syntax Tree to analyze the syntax of the source code. In addition, one uses word-embedding techniques to achieve semantic similarities, however, the approach can be affected by the removal of stop words. If developers in the project team prefer to use very long statements to express bug reports, the filter size should also be longer in the neural network settings, and these factors affect the approach. After that, DeepLocator was developed into DeepLoc by Xiao et al. The study [6] proposed The DeepLoc model is a new model based on deep learning that takes full advantage of semantic information, although the study successfully fixed the error and improved performance and accuracy of the model as shown in Figure1.

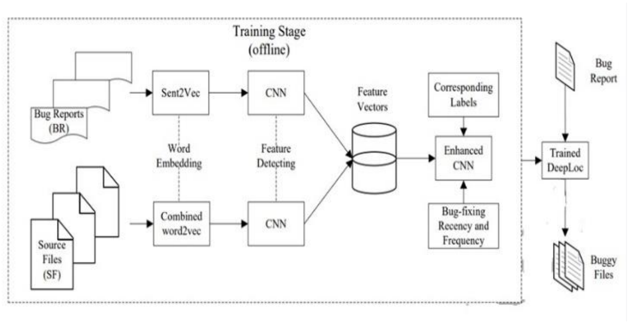

Figure 1: The Overall Structure of Deeploc

However, localization procedure made it difficult to represent the source code with Sent2Vec. Therefore, Sent2Vec is not good for converting source files into vectors because source code is written in programming language and contains reserved words such as constant, public, etc., as proven by the study [6].Furthermore, the DeepLoc model work, consisting of six phases: analysis and pre-processing, token matching, VSM similarity, stack trace, semantic similarity, and fixed bug reporting. as shows in Figure 2 This is a simplified explanation of the six stages that operate in DeepLoc:

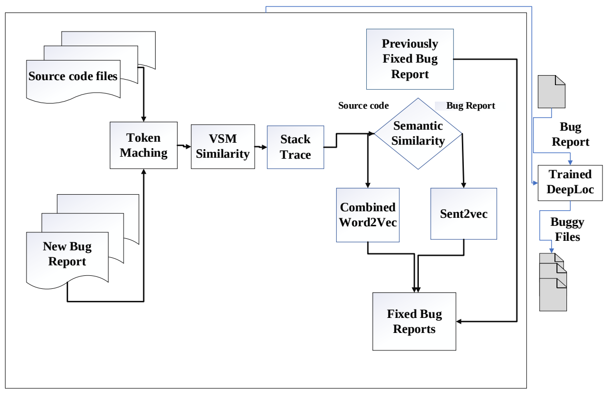

Figure 2: The Stages of The Deeploc

Some studies have demonstrated that the use of different word embedding techniques is possible in the field of error localization. There are studies that have used Word2Vec technology for source code processing and bug reporting. In addition, other studies have used Word2Vec technology for source code processing and Sent2Vec for error reporting. More importantly, studies have shown that using GloVe for both source code and bug reporting can achieve the same goal. While this study proposes to try two scenarios, the first uses GloVe technology for source code and Word2Vec technology for error reporting. As for the second scenario, that uses GloVe techniques for source code and Do2Vec for bug reporting in order to ensure the effectiveness of these proposed techniques compared to previously implemented techniques.

3. METHODOLOGY

Enhanced DeepLoc Model, a deep learning-based model that automatically detects errors in source code by compiling problematic files associated with bug reports. The model consists of six phases: analysis and pre-processing, token matching, VSM similarity, stack trace, semantic similarity, and fixed bug reporting. Moreover, in the improved approach, the change was only in the semantic similarity phase. The words themselves cannot be entered directly into CNN [13]. Thus, pre-processed words must be embedded in vectors, and there are many types of word embedding techniques that have been accepted so far in the field of bug localization and that have proven effective in this field and in different fields. In addition, this study used empirically tested techniques to convert words in bug reports and source files into vectors that preserve words with high efficiency and accuracy. Error indicators in error reports generally consist of summaries and descriptions of many words. Fortunately, error reports are written in natural language, so each word in the report is converted to a vector using one of the words embed methods known as "word2vec methods." Moreover, try to get the best results by using efficient methods to automatically handle the source code and convert the code from source to vectors quickly and accurately. Thus, the improved DeepLoc method used a semantic similarity technique known as "Global Vector." This approach is used in Spacy12's GloVe Common Crawl model to calculate semantic similarity for processing source code.

4. PROPOSED MODEL

Our proposed bug localization model utilized the Word2vec embedding techniques to process the bug report; and GloVec word embedding techniques will also be used to process the source code. as shown in the Figure 3.

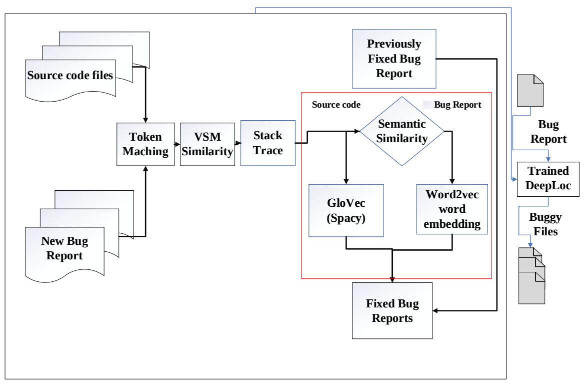

Figure 3: The Overall Structure of Our proposed bug localization model

5. RESULTS AND DISCUSSION

In this paper, the output of running the **Our proposed bug localization model** that GloVe technology was used to process the source code, while Word2Vec technology was used to process the bug report to check the effects on DeepLoc's performance in terms of the evaluation matrix: accuracy, MRR, and MAP. The dataset consists of two parts: AspectJ, which represents large projects, and SWT, which represents small projects.

PART1: RESULTS OF SWT

comparing the results of the **Our proposed bug localization model** to several complex bug localization models such as DeepLocator and DeepLoc, based on the same data set (SWT) as illustrated in Table 1.

Table 1. The Overall Performance of Scenario one and the Previous Studies on SWT

Graphics	Our Model	DeepLoc	DeepLocator
Accuracy@1	0.65	0.39	0.36
Accuracy@5	0.81	0.66	0.60
Accuracy@10	0.88	0.77	0.75
MAP	0.60	0.40	0.39
MRR	0.72	0.49	0.48

As shown in Table1, the results of our proposed bug localization model which show the best performance among all previous studies in terms of all evaluation matrix criteria. However, our proposed bug localization model gains 0.65 in Accuracy@1, Therefore, this paper found that the first scenario is effective with small projects, as illustrated in Figure 4.

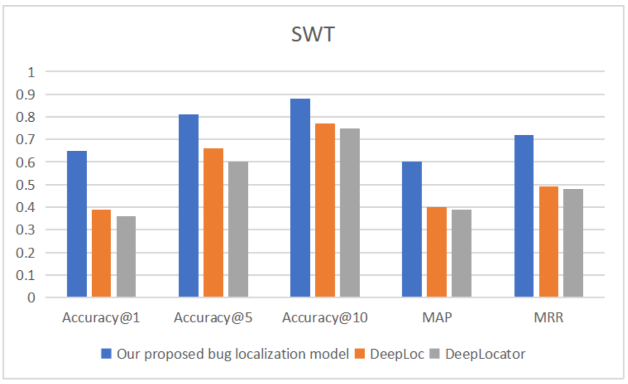

Figure 4: The Performance Comparison of Scenario 1 and the previous studies on WST

PART2: RESULTS OF ASPECTJ

Comparing the results of our proposed bug localization model to several complex bug localization models such as DeepLoc and DeepLocator, based on the same data set (AspectJ) as illustrated in Table 2.

Table 2. The Overall Performance of Scenario one and the Previous Studies on AspectJ

Graphics	Our Model	DeepLoc	DeepLocator
Accuracy@1	0.17	0.45	0.40
Accuracy@5	0.36	0.71	0.66
Accuracy@10	0.52	0.80	0.78
MAP	0.17	0.42	0.34
MRR	0.27	0.51	0.49

As shown in Table 2, the results of our proposed bug localization model on the AspectJ dataset were lower than all other models in terms of all evaluation matrix criteria. Therefore, this paper found that the first scenario is not suitable for large projects, as shown in Figure 5.

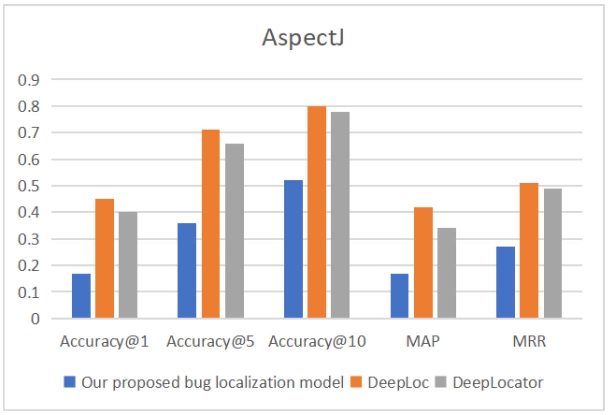

Figure 5: The Performance Comparison of Scenario 1 and the Previous Studies on AspectJ

6. CONCLUSION AND FUTURE WORK

In this paper, an approach that benefits software engineers is improved by finding the wrong part in the source code to easily correct the error later on. This has adopted the use of different word embedding methods, text analysis, and semantic similarity. This study was performed on the DeepLoc model, which is a deep learning-based model that consists of a neural network with word embedding techniques. Current approaches to error localization focus on similarities between reports and source code or relationships between term weights. However, most of these methods ignore semantic information in error reports and source files. Since there is a lexical difference between error reporting and source code in particular, the proposed approach bridges the semantic gap using embedding techniques. Keywords to remember when writing bug reports and source code In addition, compare the proposed approach with different state-of-the-art methods (DeepLocator, HyLoc, LR + WE, and Bug Locator), and then implement the approach in the AspectJ and SWT dataset projects, which contain a lot of error reports and source code files. The experimental results showed that the best performance obtained through the proposed approach was with the SWT data set.

Where the value of MAP and MRR is greater than all previous methods except for the textual properties model, whose values are slightly greater than the proposed approach. In the future, it intends to improve the performance of DeepLoc by adding a Doc2Vec technique to handle bug reports.

7. REFERENCES

[1] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn, "Bug localization using latent dirichlet allocation," *Information and Software Technology,* vol. 52, pp. 972-990, 2010.

[2] R. Gharibi, A. H. Rasekh, M. H. Sadreddini, and S. M. Fakhrahmad, "Leveraging textual properties of bug reports to localize relevant source files," *Information Processing & Management*, vol. 54, pp. 1058-1076, 2018.

[3] J. Xuan, M. Martinez, F. Demarco, M. Clement, S. L. Marcote, T. Durieux, D. Le Berre, and M. Monperrus, "Nopol: Automatic repair of conditional statement bugs in java programs," *IEEE Transactions on Software Engineering*, vol. 43, pp. 34-55, 2016.

[4] S. Khatiwada, M. Tushev, and A. Mahmoud, "Just enough semantics: An information theoretic approach for IR-based software bug localization," *Information and Software Technology*, vol. 93, pp. 45-57, 2018.

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

[6] Y. Xiao, J. Keung, K. E. Bennin, and Q. Mi, "Improving bug localization with word embedding and enhanced convolutional neural networks," *Information and Software Technology*, vol. 105, pp. 17-29, 2019.

[7] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532-1543.

[8] R. Lee, " Computer and Information Science 2021—Summer-Book " vol. Volume 985, 2021.

[9] C. Mills, E. Parra, J. Pantiuchina, G. Bavota, and S. Haiduc, "On the relationship between bug reports and queries for text retrieval-based bug localization," *Empirical Software Engineering*, vol. 25, pp. 3086-3127, 2020.

[10] S. Amasaki, H. Aman, and T. Yokogawa, "A Comparative Study of Vectorization Methods on BugLocator," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2019, pp. 236-243.

[11] J. Zhou, H. Zhang, and D. Lo, "Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports," in *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 14-24.

[12] N. Miryeganeh, S. Hashtroudi, and H. Hemmati, "Globug: using global data in fault localization," *Journal of Systems and Software*, vol. 177, p. 110961, 2021.

[13] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *arXiv preprint arXiv:1510.03820*, 2015.