SEUSL

Sri Lankan Journal of Technology

Research

SLJoT
2022 3(02) 25-29
ISSN 2773-6970

Convolution Neural Network Based Priority Prediction Approach for GitHub Issues

D.A.P. Madubashini [1] and C. Wijesiriwardana[1*]

[1]Faculty of Information Technology, University of Moratuwa

[*]Corresponding Author: chaman@uom.lk ‖ ORCID: 0000-0002-1124-425X

Received: 15-02-2022 * Accepted: 07-12-2022 * Published Online: 31-12-2022

***Abstract*—**A bug report is an important document that outlines software problems that result in unexpected errors or wrong outcomes. In large software projects, a high number of bugs are reported daily, which needs to be systematically analyzed. Predicting the priority level of reported bugs, assigning an appropriate developer, finding duplicate issues, and predicting bug resolving time are some of the critical tasks in the bug analysis process. Due to the inherent complexity of the bug analyzing process, manual bug investigation requires a significant amount of time, resources, and effort. Therefore, the need to establish automated or semi-automated approaches for assessing bug reports is extensively discussed in the literature. This research presents a novel approach to prioritize the bug reports by exploiting a Convolutional Neural Network-based approach. Furthermore, this research investigates the impact of both textual and categorical features of bug reports in improving the accuracy of priority prediction. The experiments were conducted by extracting the bug reports available in three GitHub repositories. The evaluation results confirm that the use of categorical features does not have an impact on the accuracy of the priority prediction of bug reports. Furthermore, it was observed that better prediction accuracies are shown for the datasets extracted from Bugzilla than GitHub repository.

***Keywords*—**Git issues, GitHub API, Priority prediction, Word Embedding, Convolutional Neural Network (CNN)

I. Introduction

Software developers record the bugs and the issues of software projects in bug reports as a common practice, which are stored in bug tracking systems such as Bugzilla or Jira. For instance, the Mozilla bug repository keep track of more than 670,000 bug reports with nearly 130 new bug reports add up each day [Shu *et al*., 2019]. Similarly, the Eclipse bug database contains over 250,000 bug reports with nearly 120 new bugs added each day [Uddin *et al*., 2017]. Therefore, ultimately software maintenance time and cost increase with this large number of bugs reported every day. Bug triaging is considered important in software maintenance and testing, which consists of several critical tasks such as predicting the priority level of reported bugs, assigning an appropriate developer, finding duplicate issues, and predicting bug resolving time.

Bug prioritization is vital, particularly in open-source projects because it is a measure of the responsiveness of the project [Kanwal & Maqbool, 2012]. The manual bug prioritization process is considered effort-intensive, time-consuming, and error-prone. As a result, bug reports can be assigned with incorrect priority levels [Umer *et al*., 2019]. Incorrect estimations of priority levels may result in ineffective utilization of resources, for instance by fixing less important bugs first [Almhana, R., & Kessentini]. Therefore, the need for automatic bug prioritization has been significantly investigated in the literature. Previous studies have attempted to address this issue by using various technologies like machine learning and neural networks [Baarah *et al*., 2019], [Malhotra *et al*, 2021], [Ramay *et al*., 2019]. These approaches are either based on textual features or categorical features.

In this research, we propose a Convolution Neural Network-based bug report priority prediction approach by considering both textual and categorical features. A similar study was conducted by [Umer *et al*., 2018], by only considering textual features such as 'summary' and 'description' of the bug reports. This research intends to investigate the impact of categorical features (i.e., severity, product, component, operating system, author) when used in combination with textual features in prioritizing the bug reports. The experiments are conducted on the bug reports of open-source projects found in GitHub Issues, which is a lightweight issue-tracking system that is available in all GitHub repositories. It is expected to address the following Research Questions (RQs):

RQ1: Do categorical features impact the accuracy of the priority prediction of bug reports when used in combination with textural features?

RQ2: Is it recommended to use bug reports from GitHub repository for priority prediction over the bug reports available in standard bug repositories such as Bugzilla?

The remainder of this paper is organized as follows.

Section 2 presents the Related Work of this research followed by the proposed Approach in Section 3. The Evaluation of the results is presented in Section 4. We conclude in Section 5 together with the Future works.

II. Related Work

An extensive amount of research work reported on different bug analysis approaches to predict the priority level of bug reports, including a collection of comprehensive survey reports.

The work presented by [Alenezi and Banitaan, 2013] prioritized the bug reports using two feature sets. They have evaluated the problem as a classification task using a Decision tree, Random Forests, and Naive Bayes classification algorithms. In this work, bug reports are classified into three categories: high, medium, and low and describe the feature sets under investigation. As the results, they mentioned Random Forest and Decision Tree both outperformed Naive Bayes in categorization. In terms of Precision, Recall, and F-measure, feature-set-2 outperformed feature-set-1 by a large margin. As further work they have mentioned, more research is needed to see how different combinations of characteristics affect categorization accuracy.

Another study [Kanwal and Maqbool, 2012] conducted aimed at automating the process of assigning bug priority to incoming bug reports in a bug repository, they proposed a bug priority classifier that compares the results of SVM and Naive Bayes classification methods. The experimental data show that SVM outperforms Naive Bayes in terms of overall performance. Also, when the extensive description of an issue is not included as a feature, the Naive Bayes classifier performs better.

DRONE approach aims at predicting the priority level of bug reports by using multi-factor analysis. It used a Linear regression approach to predict the priority of the bug report [Tian, 2017]. There are two parts to this approach: training and prediction. The feature extraction module and the categorization module are the two main modules. As the limitations they mentioned, collecting enough training data to create a useful prediction model is quite difficult, and its technique may suffer from the cold start problem when used on new or small projects. Also, DRONE's performance across all priority levels isn't consistent, and it could be better. For future work, they suggested building a linear regression model with only the most discriminative features and evaluating the output.

The work presented by [Choudhary and Singh, 2017] proposed a neural network-based approach for priority prediction. Their approach contains a method to obtain information from neighbor bug reports or other outside sources and used to generate an output signal that is sent to other units. They proved that the MLP-based strategy is especially effective when it comes to categorizing distinct priorities. But this study only employed five versions and three products of the Eclipse project. As the future work, they suggest that different Eclipse products and versions, as well as Mozilla and other third-party products, be used, and that cross-components be applied by building an Eclipse global dictionary

III. Approach

Figure 1 presents the high-level architecture of the proposed approach to prioritize the bugs that are extracted from Github, which contains four main subsections: feature extraction, pre-processing, word embedding, and priority prediction.

A. Data and Feature Extraction

The priority is an important feature of a bug report because it indicates how quickly it should be fixed. Bug reports are frequently sent, either with an inaccurate priority level or without stating a priority level. Developers manually repair or assign the priority of each issue report after reading them. Manually prioritizing bug reports necessitates knowledge and resources. As per the literature, it was decided to utilize the following discriminative features on bug reports in GitHub [Tian *et al*., 2013].

Severity: Basically, severity determines the customers, whereas developers determine priority. When a developer assigns a priority level to a bug report, the severity stated by users has an impact, but it's not the only factor to consider. For example, while a bug being corrected may be a serious issue for a specific reporting, it's possible that the eclipse team shouldn't fix it.

Author: If there is a pattern as an author consistently reports high-priority problems, he or she may continue to do so in the future. Furthermore, the more defects a user reports, the more accurate his or her bug severity assessment will be.

Priority: The priority of a bug report/issue defines how soon to fix it. High priority will be assigned for the critically important bugs whereas low priority will be given otherwise.

Related report: This feature shows the possibility of having the same priority for similar issues. It calculates by comparing the textual features.

Sentiment value: Calculate the emotion of each bug report to categorize whether the reporter's emotion is good or negative

B. Pre-processing and Word Embedding

Irrelevant and unwanted text, such as punctuation, can be found in bug reports. Feeding irrelevant text into classification algorithms is an overhead, since it lengthens processing time and consumes more memory. NLP techniques are used to preprocess git issues such as tokenization, stop-word elimination, negation handling, spell correction, modifier word recognition, word inflection, and lemmatization are popular NLP techniques used to preprocess git issues.

Next, a vector for each git issue is created. We feed the preprocessed words w1, w2, ..., wn into a word embedding model. Bag Of Word (BOW) is the model that creates the vocabulary, a list of words that occurred in the git issue

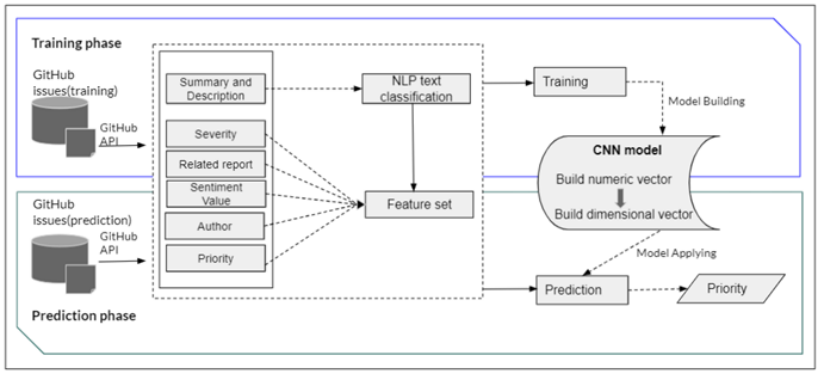

Figure 1: High-level Architecture of the Proposed Approach

description, where each word has its own index. This makes it possible to create a vector for each summary. Then we count each occurrence in the vocabulary using the summary we wish to vectorize. The resulting vector will contain the vocabulary's length as well as a count for each word in the vocabulary. A feature vector is the product of this process. Each dimension of a feature vector can be a numeric or category feature. Each word is represented as a vector in the word embedding model. Text can be vectorized in a variety of ways, including:

- Each word is represented as a vector.
- Each character is represented as a vector of characters.
- N-grams of words or characters (N-grams are overlapping groupings of numerous succeeding words or characters in the text) depicted as a vector

Unlike one-hot encoding, which is hardcoded, this approach represents words as dense word vectors (also called word embeddings) that are trained. This implies that word embeddings acquire more data in fewer dimensions.

C. Priority Prediction

An autonomous approach is used for bug report priority prediction, where a particular priority level is assigned for each bug report. The proposed approach uses textual features together with a set of categorical features such as severity, author, the priority of related report and sentiment value in the CNN. The selection of CNN is based on the following reasons: CNN uses the vector concatenation method to concatenate incoming inputs into one long input vector, CNN layers may learn the deep semantic relationships between input words, CNN significantly reduces training time due to its capability for parallel computation on modern powerful GPU and CNN may use different filter size filters that avoid the exploding gradient problem of recurrent neural networks.

IV. Evaluation

Git issues are selected from two GitHub repositories (i.e., Angular and WSO2 Product-IS) and one Bugzilla repository (i.e., Eclipse). The extracted GitHub issues labelled as "bug" from GitHub repositories using GitHub API and Node.js libraries. We've only included GitHub issues that are marked as CLOSED since their priorities have been mentioned. Then fetched the above-mentioned attributes from each bug and added this data to a CSV file. Table I shows the number of git issues extracted from each repository.

Table I: Number of issues

Repository name	Number of issues
Eclipse	3,060
Angular	1,081
WSO2 product IS	449

Table II: Priority distribution

Priority level	Eclipse	WSO2 Product IS	Angular
P1	0	102	0
P2	281	0	82
P3	2,139	14	540
P4	53	127	459
P5	13	201	0

There are five class labels namely P1, P2, P3, P4, and P5. Table II shows the distribution of git issue priority levels in each repository.

The evaluation process is conducted as follows. To answer RQ1, first, bug reports are extracted from GitHub repositories (i.e., WSO2 Product-IS and Angular) and applied NLP techniques to preprocess the text features of bug reports. Then partition the dataset as a training and testing dataset. The training part is used to train the CNN model and the testing part remains until the testing phase. Given a bug report, its performance is evaluated by using specific precision, recall, and F1-score.

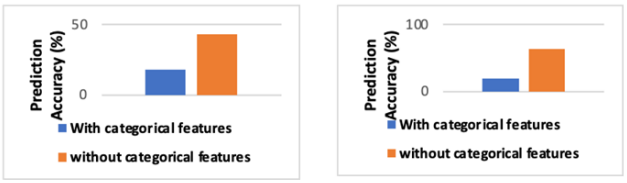

Figure 2: Priority prediction accuracy with and without categorical feature for WSO2 Product-IS (on the left) and Angular (on the right)

Figure 2 shows the proposed CNN based priority prediction gives 43%, and 63% accuracies respectively for WSO2

Product-IS and Angular without considering the categorical features such as severity, the priority of the related report, author, and sentimental value. However, when it is used with the categorical features, the prediction accuracy decreases to 18% and 19% respectively. Thus, in answering RQ1, it is evident that the inclusion of the categorical features does not increase the prediction accuracy, hence it is not recommended to use categorical features in combination with textual features as per the experimental results. Furthermore, we compared our approach with the previous work reported by Umer et. al [9] to validate the proposed CNN based priority prediction model in terms of precision, recall and F1-score. As presented in Figure 3, it was evident that both approaches perform at more or less a similar accuracy.

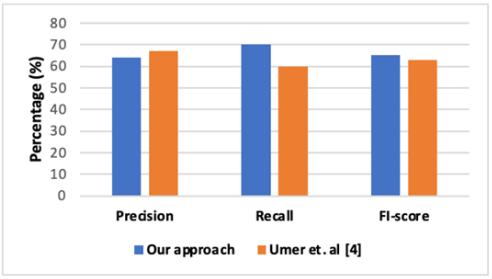

Figure 3: Comparison with an existing approach reported by Umer et. al [4]

To evaluate RQ2, the experiment was conducted for Eclipse bug reports that are extracted from the Bugzilla database. Based on the experimental results, as shown in Figure 4, it was observed that the accuracies of GitHub repositories are quite less than the Bugzilla repository. Thus, it is recommended to use standard bug repositories such as Bugzilla in predicting the priority of bug reports. However, to overcome the situations where the researchers do not have access to standard commercial bug tracking systems, it is recommended to make use of GitHub issues by further fine-tuning our experimental approach.

Figure 4: Prediction Accuracy of Bugzilla Repository (Eclipse) Vs GitHub Repository (Angular and WSO2 Product-IS)

V. Conclusion and Future Work

This research intended to answer two important research questions to experimentally find out the impact of the categorical features in predicting the priority of bug reports and to observe the suitability of different issue tracking repositories for the same purpose. The experiments were conducted on bug reports extracted for three software systems (i.e., Angular, Eclipse, WSO2 Product-IS) selected from two repositories (i.e. GitHub and Bugzilla). As per the evaluation results, it was observed that the introduction of categorical features does not impact the prediction accuracy. Rather, it shows better prediction accuracies when not using categorical features together with textual features. Besides, as per the second experiment, it was noted that better prediction accuracies are shown for the datasets extracted from Bugzilla than GitHub repository.

Since CNN gives the semantic relationship between features, it is difficult to find relationships between numerical features. Therefore, as further work, it is expected to investigate a regression-based approach by further utilizing new features such as bug time-stamps, product names.

References

Alenezi, M., & Banitaan, S. (2013, December). Bug reports prioritization: Which features and classifier to use?. In 2013 12th International Conference on Machine Learning and Applications (Vol. 2, pp. 112-116). IEEE.

Almhana, R., Kessentini, M. (2021). Considering dependencies between bug reports to improve bugs triage. *Automated Software Engineering, 28*(1), 1-26.

Baarah, A., Aloqaily, A., Salah, Z., Zamzeer, M., & Sallam, M. (2019). Machine learning approaches for predicting the severity level of software bug reports in closed source projects. *International Journal of Advanced Computer Science and Applications, 10*(10.14569).

Choudhary, P. A., & Singh, S. (2017). Neural Network Based Bug Priority Prediction Model using Text Classification Techniques. *International Journal of Advanced Research in Computer Science, 8*(5).

Kanwal, J., & Maqbool, O. (2012). Bug prioritization to facilitate bug report triage. *Journal of Computer Science and Technology, 27*(2), 397-412.

Malhotra, R., Dabas, A., Hariharasudhan, A. S., & Pant, M. (2021, January). A study on machine learning applied to software bug priority prediction. In 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 965-970). IEEE.

Ramay, W. Y., Umer, Q., Yin, X. C., Zhu, C., & Illahi, I. (2019). Deep neural network-based severity prediction of bug reports. *IEEE Access, 7*, 46846-46857.

Shu, R., Xia, T., Williams, L., & Menzies, T. (2019). Better security bug report classification via hyperparameter optimization. arXiv preprint arXiv:1905.06872.

Tian, Y. (2017). Mining software repositories for automatic software bug management from bug triaging to patch backporting.

Uddin, J., Ghazali, R., Deris, M. M., Naseem, R., & Shah, H. (2017). A survey on bug prioritization. *Artificial Intelligence Review, 47*(2), 145-180.

Umer, Q., Liu, H., & Illahi, I. (2019). CNN-based automatic prioritization of bug reports. *IEEE Transactions on Reliability, 69*(4), 1341-1354.

Umer, Q., Liu, H., & Sultan, Y. (2018). Emotion based automated priority prediction for bug reports. *IEEE Access, 6*, 35743-35752.