Using CNN to Predict the Resolution Status of Bug Reports

Muhammad Ali Arshad¹ and Huang Zhiqiu²*

¹Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
²Key Laboratory of Safety-Critical Software, NUAA, Ministry of Industry and Information Technology, Nanjing 211106, China

*Email: m.aliarshad108@nuaa.edu.cn; zqhuang@nuaa.edu.cn

Abstract. Bug tracking systems (BTS) are a resource for receiving bug reports that help to improve software applications. They usually contain reports reported by the end-users or developers. Bug Reports contain some suggestions, complaints, etc. The problem is that every submitted bug report is not accepted to implement. Mostly bug reports are rejected because they are incomplete, duplicate, expired, etc. while only a few are accepted to implement. But the developers have to check every bug report manually that needs many resources (i.e. labour, money, time). In this study, we proposed a convolutional neural network (CNN) based approach to automatically classify bug reports as accepted and rejected. Results show that the proposed approach achieves the highest performance as compared to closely related works.

1. Introduction
There are different BTS (i.e. JIRA [1], Redmine [2], Mozilla [3], etc.) which are used by software applications (i.e. Firefox, Calendar, SeaMonkey, etc) to track bugs (i.e. enhancement, defect, and task). These bugs are used for the improvement of software applications. Enhancement reports are commonly used for future releases. New versions of an application are based on the fulfillment of additional feature requirements, change in existing features, or solution of issues that were being faced by the users. Analysis of bug reports extracted for different applications extracted from Bugzilla-Mozilla BTS from 1997-09-10 to 2016-07-13 shows that 21.75% of bug reports are accepted and 79.25% of bug reports are rejected. But the developers need to check every report manually to make sure that they did not miss any useful bug report. Because bug reports are very useful for the success of software applications. Bug reports capture the user requirements and every company builds applications for its users. Users are the main assets for the software companies. So, the timely implementation of bug reports is necessary for the success of software applications. Most of the bug reports are rejected because of different reasons such as duplicate, incomplete, expired, etc. Especially, there are two issues that can be occurred while manually resolution prediction.

1.1. Triage Workload
Every day, several bug reports are reported to BTS. For example, Eclipse and Bugzilla receive an average of 120 and 170 bug reports daily, respectively [4]. To check all the bug reports manually place a heavy burden on the bug triage owners, bug fixer, and quality assurance team.

1.2. Inaccurate Resolution Prediction
Manually resolution prediction for each enhancement report may lead to errors. For example, useless bug reports may be accepted and useful enhancement reports may not be accepted and implemented in the system that is very important to compete for the market. New versions of an application are based on the enhancement reports. Hence, accurate resolution prediction of software enhancement reports is a very important task in the bug management process. Existing methods used by the BTS for bug triage is still not an automatic process, empirical studies show that when there are a number of issues reported by the users than it is very expensive, and chances of inaccuracy are high. Like other factors such as severity prediction, priority prediction, and fixer assignee. Resolution prediction of bug reports is a very important factor to automate the bug triage process for the timely implementation of bug reports. Previously proposed approaches for bug resolution prediction focus only on traditional machine learning algorithms, e.g., Support Vector Machine, decision tree, etc. However, recently deep learning-based approaches have been approved to achieve state-of-the-art performance. They have enhanced results significantly in different fields e.g., speech recognition [5], sentiment analysis [6], and computer vision [7]. Deep learning has many benefits over machine learning-based approaches as it provides many effective, efficient, and popular models, the previously proposed approaches have not used deep learning approaches for bug resolution prediction. Various predictive models have been proposed by researchers [8, 9] to predict the resolution status of newly submitted bug reports. The idea behind the proposed approach differs from already existing approaches as it consists of using bug report attributes such as summary and description. Moreover, we used effective pre-processing techniques, we used contractions for dimensionality reduction and customized stop-words removal. We removed prepositions from the features matrix except which are useful (i.e. should, could, must). We used the convolutional neural network for classification. We are the first to implement an automatic bug resolution prediction system using deep learning. And we are the first one to predict the resolution status of bug reports (Enhancement, task, and defect). Previously proposed approaches used just enhancement reports. They did not use tasks and defects for their approaches. Results for the proposed approach show that the proposed approach performs state of the art than other existing approaches. The problem of predicting the resolution status of the enhancement reports is formulated as a supervised text classification task. We describe the major contributions of our work as follow:

An automated CNN based approach for the resolution prediction of bug reports. We use summary and description of bug reports to feed a neural network to predict the resolution status of newly submitted bug reports.

- Using this application limited human resources are needed to inspect a lot of useful enhancement reports from a lot of users.
- Developers can respond more quickly to likely to be approved enhancement reports instead of spending much time to check each enhancement report manually.
- Reporters can use this system to improve their enhancement reports. They can use to check the prediction of the enhancement reports to be accepted or rejected and can they can improve the quality of the enhancement reports before submission.
- Evaluation results of the proposed approach show that the proposed approach is accurate in the resolution prediction of bug reports.

The remainder section of this work is structured as follows: Section 2 describes the related work. Section 3 describes the background knowledge of BTS. Section 4 describes the methodology for the proposed approach. Section 5 discusses, evaluates the results, and make the comparison with already existing approaches. Section 6 concludes this work and propose future work

2. Related Work

In this section, we discuss existing studies to predict the resolution status of bug reports. Nizamani et al. [8] extracted the data of enhancement reports from Bugzilla-Mozilla BTS. They used just summary of enhancement reports to predict the resolution status of the enhancement reports and adopt a binary classification approach. They use Multinomial Naive Bayes and VSM (TF-IDF). Umer et al. [9] use the same dataset extracted by Nizamani et al. [8]. They calculate the sentiment score of the summary
using SENTI4SD, a tool for sentiment score calculation for software engineering documents, and add sentiments with summary features. They adopt Support Vector Machine with VSM (TF-IDF) for the automatic resolution prediction of software enhancement reports. Accuracy of both Nizamani et al. [8] and Umer et al. [9] approaches are considered for the automatic resolution prediction. To the best of our knowledge, these two works are considered state-of-the-art and similar to our target task. The proposed approach is different from their approaches as the proposed approach identifies additional vital features that were previously ignored and take advantage of the deep learning approach. Most of the state-of-the-art classification approaches used for bug related document classification focus mainly on automated fixer assignee, severity prediction, priority prediction, bug fixing time prediction, duplicate issue prediction, and issue classification. Ardimento et al. [10] used BERT to predict the fixing time of a bug as slow and fast. Defect, enhancement, and task are three types of bugs in most BTS systems. Sometimes, reporters do not have enough knowledge about the type of bug and developers have to manually check the type of each bug and separate bugs from non-bugs. To manually check and classify the bugs to correct their type is a time-consuming task. Antoniol et al. [11] proposed an approach to automatically classify bugs. Later, Zhou et al. [12] combined text-mining and data mining techniques to classify bugs, they used both structured and unstructured text. Campbell et al. [13] used ABLSTM to classify issues as bugs and tasks. Tao Zhang et al. [14] and Kumari et al. [15] also used both structured and unstructured text for the severity prediction and fixer recommendation. Duplicate bug reports occur when the reports with the same description are submitted by more than one user or developer. There are several works done [16-18] on duplicate bug detection. In this study, we analyze the impact of different features as well as the impact of different machine learning and deep learning approaches on the performance of automatic resolution prediction.

3. Background Knowledge of BTS

In our work, we propose a CNN based approach to predict the resolution of newly submitted bug reports based on similar historical bug reports, their features and resolution status. Thus, in this section, we introduce some background knowledge about bug reporting and tasks in bug resolution.

3.1. Tasks in Bug Resolution

When a new bug report is submitted, the developers (i.e., triage owner, bug fixer, QA team, etc.) work together for resolving the bug. Figure 1 shows the bug life cycle for a bug in Bugzilla-Mozilla BTS. The period between the bug is detected until the point the bug is fixed is known as the bug lifecycle. The first stage of the new bug report is ‘UNCONFIRMED’, someone in the quality assurance team tries to reproduce it, after the confirmation, its status changes to ‘NEW’. During this process, the triage owner assigns priority, severity, resolution, and assign it to an appropriate assignee. Later, the stage changes from ‘NEW’ to ‘ASSIGNED’. Once the assignee completes the bug-fixing task, the status changes to ‘RESOLVED’. If the assignee fails to fix it, it is marked as ‘NEW’ again and re-triaged again. It is known as bug reassignment. Once the bug is ‘FIXED’, the state changes to ‘CLOSED’, and the resolution status is ‘FIXED’. Subsequently, if the developer finds that the bug is not fixed according to the reporter’s requirements, or the reporter finds the bug is not resolved appropriately. It can be reopened and the cycle is re-executed. Since resolution prediction is a very important task for triager as well as a fixer. In this work, we focus on developing a new approach to predict the resolution of bug reports.

3.2. Bug Reports’ Resolution

Bug report’s resolution status describes either the bug report has been accepted or not. The resolution has a special field in bug reports. There are different possible types of bug resolution for enhancement reports in Bugzilla-Mozilla. They include ‘FIXED’, ‘INVALID’, ‘DUPLICATE’, ‘WORKSFORME’, ‘EMPTY’, ‘WONTFIX’, ‘INCOMPLETE’, and ‘EXPIRED’. Table 1 briefly explains each resolution field. As only ‘FIXED’ are accepted and all others are rejected. Therefore, we deal with it as a binary classification by reducing the classes into binary. The total reports for the dataset used in our
experiments is shown in Figure 2 with the percentage of acceptance and rejection in each application for top 10 applications from Bugzilla-Mozilla. Since, only reports with resolution status ‘FIXED’ are accepted, all others are rejected. Therefore, we consider ‘FIXED’ as accepted and all other rejected.

Table 1. Resolution types with a brief description.

| Resolution  | Definition                                                                 |
|-------------|-----------------------------------------------------------------------------|
| FIXED       | FIXED describes that a fix for the bug has been checked and the bug can be fixed. |
| INVALID     | INVALID resolution is given to the bug reports that show the described requirement is not a bug. |
| WONTFIX     | WONTIFX is assigned to bug reports that describe the requirement as a bug that will never be implemented. A bug that doesn't affect any actual user |
| MOVED       | MOVED describes that the bug is now being worked on another BTS.             |
| DUPLICATE   | Duplicate indicates the described bug report is a duplicate of an existing bug report. |
| WORKSFORME  | It indicates that all the attempts to recreate the bug have been done by someone in the QA team. But all the attempts are unsuccessful. |
| INCOMPLETE  | INCOMPLETE resolution is assigned to the bug reports that are described vaguely and do not tell how to reproduce them. |
| EXPIRED     | Expired is assigned to the bug reports that have insufficient data.          |
| EMPTY       | No resolution status is given.                                              |

4. Methodology
This section explains the steps involved in the proposed approach used to predict the resolution status of enhancement reports. The proposed approach predicts the resolution of the enhancement reports as follows. First, we extract the data of the top ten applications from Bugzilla-Mozilla BTS using the REST API. Second, we apply different pre-processing techniques using NTLK and python libraries (i.e. pycontractions, spellchecker). Third, we create a vector of each report using Word2Vec (Skip-gram) model. Fourth and final, we train CNN classifier and machine learning classifiers for resolution prediction. We input the feature vector consists of vital features from enhancement reports for the automatic resolution prediction of enhancement reports.
4.1. Dataset

we extracted the dataset from Bugzilla-Mozilla BTS. We extract the dataset of ten applications from the duration 1997-09-10 to 2016-07-13. These are the top ten applications that use Bugzilla-Mozilla BTS for tracking bugs. Figure 3 shows the number of rejected bug reports for every application is much higher than the accepted bug reports. The proposed approach can help to save resources by automatically predicting the resolution status of bug reports. These are 42423 bug reports in total from which only 21.75% of bug reports are accepted while 79.25% of bug reports are rejected because they are either a duplicate of an existing bug report, do not have complete information that how the bug occurred and what was the result of the bug, the work is being on the same bug in another BTS, etc.

Figure 2. Accepted and Rejected Enhancement reports for each Software Application.

4.2. Preprocessing

We use pre-processing to remove unwanted data which is not necessary and useful for the proposed approach. We perform contractions, tokenization, lowercase-conversion, stop-word removal, word inflection, and lemmatization to clean the summary and description of the enhancement reports. We used Contractions to convert the contractions into normal words, tokenization to break the sentences into the words, lower case conversion to convert all the words into lower case, Bugzilla stops word removal to remove only those stop-words that are not helpful for our approach. We did a survey to ask about the useful words in the bug reports. We did not remove all the words from bug reports while we remove only the stop-words not necessary for the proposed approach. Later, we used word inflection to singularize the words and lemmatization to convert the word into a simple form (positive degree).

4.3. Classifier

We used CNN for resolution prediction of bug reports because it may learn the semantical relationship between the words of bug reports. The configuration of our CNN classifier is shown in Figure 4. Because machines cannot understand and interpret natural language, we need to convert it into a format that machines can understand. For this purpose, we convert the sentences into a k-dimensional feature vector. We used the embedding layer as the first layer of our model to turn indexes into a vector of fixed size length. We feed the numerical vectors into CNN, we used three layers of CNN as follows, filter=64, kernel size=1, and activation=tanh. We feed the output of CNN to the max. pooling layer to select the maximum values from the input layer for feature reduction. In the last, we use a Fully-connected layer and sigmoid function that outputs the probability between 0 and 1. We use Adam as an optimizer and binary_crossentropy to measure the performance of our proposed binary classification model.
4.4. Metrics
To evaluate the performance of classification algorithms the most commonly used metrics are accuracy, precision, recall, and f-measure. Therefore, we used accuracy, precision, recall, and f-measure to evaluate the performance of the proposed approach.

\begin{align*}
\text{Accuracy} &= \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (1) \\
\text{Precision} &= \frac{TP}{(TP + FP)} \quad (2) \\
\text{Recall} &= \frac{TP}{(TP + FN)} \quad (3) \\
\text{F-measure} &= \frac{2 \times (\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (4)
\end{align*}

Where Accuracy, Precision, Recall and F-measure is the accuracy, precision, recall and f-measure of the classifier in predicting the resolution status of bug reports. TP is the total number of bug reports whose resolution status is correctly predicted. TN is the number of bug reports whose resolution is incorrectly predicted as rejected. FP is the total number of bug reports whose resolution is correctly predicted as rejected. FN is the total number of bug reports incorrectly predicted as approved.

5. Evaluation

5.1. Performance of the Proposed Approach
Table 2 and Figure 4 show the performance of the proposed approach on top ten applications. The adoption of features from the description and summary, sentiments of the summary to train a deep learning-based classifier, seems more effective than the already existing approaches. However, unlike previous approaches described in the related work. In this work, it was thought to take advantage of deep learning approaches for text classification and recent studies ignored the description of the enhancement report and deep learning-based approaches. We used additional features from an enhancement report. Results show that description features are also very important in enhancement report resolution prediction. Toolkit achieved the highest accuracy of 85.51% and a recall of 86.06%.
5.2. Performance of the CNN Based Approach against State-of-the-art Approaches

Table 3 and Figure 5 show that shows the comparison result of the proposed approach against state-of-the-art approaches. The proposed approach achieves an average accuracy of 78.65%, average precision of 74.98%, an average recall of 78.66%, and an average f-measure of 75.86% and proved best as compared with Nizamani et al. [8] and Qasim et al. [9] approaches.

| Approach          | Accuracy | Precision | Recall | F-Measure |
|-------------------|----------|-----------|--------|-----------|
| Nizmani et al. [8]| 65.23%   | 63.55%    | 65.55% | 61.29%    |
| Qasim et al. [9]  | 73.72%   | 72.06%    | 74.05% | 72.63%    |
| Proposed          | 78.65%   | 74.98%    | 78.66% | 75.86%    |

Table 2. Performance of the CNN based approach.

| Application       | Accuracy | Precision | Recall | F-Measure |
|-------------------|----------|-----------|--------|-----------|
| Bugzilla          | 77.50%   | 72.87%    | 77.02% | 73.97%    |
| Calendar          | 76.90%   | 72.13%    | 75.97% | 73.20%    |
| Camino Graveyard  | 79.36%   | 75.85%    | 79.55% | 76.87%    |
| Core              | 74.67%   | 72.69%    | 74.51% | 73.12%    |
| Core Graveyard    | 75.90%   | 73.56%    | 76.46% | 74.10%    |
| Firefox           | 82.65%   | 80.55%    | 82.98% | 81.33%    |
| Mail News Core    | 74.83%   | 68.36%    | 74.59% | 69.38%    |
| SeaMonkey         | 85.20%   | 82.32%    | 84.02% | 83.05%    |
| Thunderbird       | 73.87%   | 70.56%    | 75.50% | 70.97%    |
| Toolkit           | 85.51%   | 80.95%    | 86.06% | 82.70%    |
| Average           | 78.65%   | 74.98%    | 78.66% | 75.86%    |
Figure 5. Comparison of the proposed approach with state-of-the-art approaches.

5.3. Comparison with Traditional Machine Learning-based Approaches
We compared the proposed approach with alternative machine learning approaches such as logistic regression (LR), random forest (RF), multinomial naive Bayes (MNB), k nearest neighbors (KNN), and decision tree (DT) to make a performance comparison between the proposed approach and machine learning approaches. Table 4 and figure 6 show the performance of the proposed approaches against machine learning approaches. Results show that the proposed approach achieves highest accuracy, precision, recall and f-measure to all the machine learning-based approaches. The proposed approach outperforms the LR, RF, MNB, SVM, KNN, and DT and obtains the best accuracy, precision, recall, and f-measure.

Table 4. Performance of the proposed approach.

| Models | Accuracy | Precision | Recall | F-measure |
|--------|----------|-----------|--------|-----------|
| LR     | 62.01%   | 59.25%    | 61.62% | 58.38%    |
| RF     | 64.31%   | 62.21%    | 65.46% | 61.27%    |
| MNB    | 63.02%   | 60.90%    | 63.07% | 60.28%    |
| SVM    | 69.46%   | 67.85%    | 69.48% | 68.03%    |
| KNN    | 71.01%   | 70.15%    | 72.22% | 68.65%    |
| DT     | 76.28%   | 72.27%    | 74.90% | 72.71%    |
| Proposed | 78.65% | 74.98%    | 78.66% | 75.86%    |

Figure 6. Comparison of the proposed approach with traditional machine learning approaches.

6. Conclusion and Future Work
The goal of this research is to find the resolution of bug reports automatically, using some information available in bug reports, such as a summary and description of the bug report. The data was extracted
from Bugzilla-Mozilla, text pre-processing was necessary for dimensionality reduction, to reduce the data noise. Resolution status has been discretized into accepted and rejected. Finally, the data was used to train a deep learning-based classifier. Results show that the proposed approach is effective in resolution prediction. An accuracy of 78.65%, a precision of 74.98%, a recall of 78.66%, and an f-measure of 75.86% has been achieved, which is significantly enhanced than the closely related works. Future work includes implementing a system for bug resolution prediction using BERT. BERT generates contextualized embedding and it preserves the context of words within sentences. Context is also important in automatic resolution prediction.

Acknowledgements
This work is supported by the National Key Research and Development under Grant No. 2018YFB1003902. We are highly thankful to the reviewers for their useful comments.

References
[1] Atlassian 2020 Jira issue tracker. https://www.atlassian.com/software/jira/, Sep 2020
[2] Redmine Issue Tracker 2020 https://www.redmine.org/, Sep 2020
[3] Mozilla. Bugzilla issue tracker. https://bugzilla.mozilla.org/, Sep 2020
[4] Guo P J, Zimmermann T, Nagappan N, and Murphy B 2010Characterizing and predicting which bugs get fixed: an empirical study of Microsoft Windows,” in Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering 1 pp 495–504.
[5] Graves A M and Hinton G 2013 Speech recognition with deep recurrent neural networks,” in 2013 IEEE international Conference on Acoustics, Speech and Signal Processing pp 6645–6649
[6] Sharma M, Bedi P, Chaturvedi K K, and Singh V B 2012 Predicting the priority of a reported bug using machine learning techniques and cross project validation in 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA) pp 539–545
[7] Ouyang X, Zhou P, Li C H, and Liu L 2015 Sentiment analysis using convolutional neural network in 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic And Secure Computing; Pervasive Intelligence And Computing pp 2359–2364
[8] Nizamani Z A, Liu H, Chen D M, and Niu Z 2018 Automatic approval prediction for software enhancement requests Autom Softw Eng 25(2) 347–381
[9] Umer Q, Liu H, and Sultan Y 2019 Sentiment based approval prediction for enhancement reports J Syst Softw, 155 57–69
[10] Ardimento P and Mele C 2020 Using BERT to Predict Bug-Fixing Time in 2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS) pp 1–7
[11] Antoniol G, Ayari K, Di Penta M, Khomh F, and Guéhéneuc Y-G 2008 Is it a bug or an enhancement? A text-based approach to classify change requests in Proceedings of the 2008 Conference of The Center for Advanced Studies on Collaborative Research: Meeting of Minds pp 304–318
[12] Zhou Y, Tong Y, Gu R, and Gall H 2016 Combining text mining and data mining for bug report classification J Softw Evol Process 28(3)150–176
[13] Campbell D, Cabrera-Diego L A, and Korkontzelos Y 2020 What is the Message About? Automatic Multi-label Classification of Open Source Repository Messages into Content Types in Joint European-US Workshop on Applications of Invariance in Computer Vision pp 520–531
[14] Zhang T, Chen J, Yang G, Lee B, and Luo X 2016 Towards more accurate severity prediction and fixer recommendation of software bugs J Syst Softw 117 166–184
[15] Moran K 2015 Enhancing android application bug reporting Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering pp 1045–1047
[16] Runeson P, Alexandersson M, and Nyholm O 2007 Detection of duplicate defect reports using natural language processing 29th International Conference on Software Engineering (ICSE’07) pp 499–510
[17] Jalbert N and Weimer W 2008 Automated duplicate detection for bug tracking systems,” in 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN) pp 52–61

[18] Neysiani B S and Babamir S M 2020 Automatic Duplicate Bug Report Detection using Information Retrieval-based versus Machine Learning-based Approaches