A Bug Triage and Localization Technique based on Bug Reports

Classification

Tao Zhang
School of Computer Science, The University of Seoul, Seoul, Korea
kerryking@ieee.org
Advisor: Byungjeong Lee

Abstract: With a great number of software products that have been developed, bug fixing is difficult due to a large number of submitted bug reports each day. Sometimes developers usually describe the same errors in the different bug reports, these bug reports are called duplicate bug reports, the increasing number of duplicates lead to a large amount of time and effort for identifying and analyzing bug reports. In addition, recommending appropriate developers to fix bugs (bug triage) and finding the locations of bugs in the source code files (bug localization) have become the primary problems in software maintenance. However, a sizeable number of reassignments and manual bug localization lead to increased time for fixing bugs. In order to resolve these problems, in this paper, I propose a novel method for executing bug triage and localization based on bug reports classification. The experimental results on two large-scale open source projects including JBoss and Eclipse show that the method can effectively classify bug reports to carry out bug triage and localization.

Keywords: duplicate bug reports, bug triage, bug localization, social network, SVM, taxonomy, smoothed UM, probability model, experience model

1. Introduction

As software products have become increasingly large and complex, it is difficult to carry out software maintenance, especially in bug fixing for large-scale software systems. Such software maintenance activities rely on bug reports to correct defects in source code files.

However, for large scale software systems, a large number of bug reports are submitted to the related bug repositories each day. The workload of developers is increasing when manually handle these bug report, especially in duplicate bug reports [1]. In addition, bug triage [2] and bug localization [3] have become the primary challenges for software maintenance. Towards bug triage, if the bug is not assigned to an appropriate developer, it must be reassigned to another fixer for correcting. A sizeable number of bug reassignment decreases the probability of the bugs being fixed and increases the fixing time. With regard to bug localization, it is often costly to manually locate the source code files to be changed based on the given bug report.

In order to resolve these problem, it is necessary to classify bug reports so that supporting bug triage and bug localization. In this study, I propose a novel way to complete the goal. First of all, I adopt Support Vector Machine (SVM) to discriminate duplicates and utilize taxonomy algorithm to arrange these bug reports to appropriate bug types according to pre-defined bug rules. Based on the classification results, for bug triage, I retrieve the similar bug reports by utilizing smoothed Unigram Model (UM) [4] instead of traditional Vector Space Model (VSM) when a new bug comes, next I propose a new algorithm combining probability model and experience model for ranking candidate developers so that recommending the best developer for fixing a new bug. For bug localization, I rank the source code files of bug reports in the list of similar bug reports to the given bug.

2. Method

The proposed approach includes three phases: bug reports classification, similar bug reports retrieval and bug triage and localization.

In the phase of bug reports classification, the main task is to train a model via SVM for discriminating duplicate bug reports. Afterwards, I classify the bug reports by using taxonomy technique according to pre-defined bug types and corresponding bug rules [5].

When a new bug comes, an Information Retrieval (IR) based technology is executed based on the texture similarity of a new bug and historical bug reports in the bug reports database. I utilize smoothed UM to represent the bug reports as vectors, and compute the texture similarity using KL divergence [6].

After that, by utilizing these features, on the one hand, I construct a social network to analyze the probability of fixing the given bug for each candidate developer to build the probability model, and I get the number of fixed bugs, the activity factors and fixing cost for each candidate to construct the experience model. Then I combine two models as a hybrid bug triage algorithm for recommending appropriate developers to fix the new bug; on the other hand, I rank the extracted source code files by computing the similarity between the new bug and these files for locating the files where the new bug should be fixed.

3. Experiment

I collect 964 and 1122 bug reports from JBoss and Eclipse, respectively. For each bug report, I extract the corresponding features such as the assignees and commenters, the summary, the description, the number of comments, relevant source code files, etc.

I conduct the experiments by executing data corpus creation, query construction, bug reports retrieval and ranking process. In the process of data corpus creation, after preprocessing all extracted bug reports, each one is arranged to
the corresponding class and denoted as "duplicate" or "non-duplicate." For query construction, I collect the summary and description of each testing bug report as a query to retrieve similar bug reports in the data corpus. In the bug reports retrieval process, this process is based on the KL divergence between the query and each bug report in the data corpus. Ranking process is responsible for ranking candidate developers and relevant location files of similar bug reports to the new bug.

![Fig. 1 Evaluation of Bug Triage Methods in JBoss and Eclipse](image)

Fig. 1 shows the evaluation results. It can be seen that the bug triage algorithm based on the classification result for bug reports shows better performance than other methods due to having the highest F-measure. Furthermore, the performance of DRA [7] outperforms DREX [8] with two metrics (Frequency and OutDegree) and SVM-based Recommendation [9].

Table. 1 Comparison of Bug Localization Models

<table>
<thead>
<tr>
<th>MRR(%)</th>
<th>JBoss</th>
<th>μ=0.4</th>
<th>tf-idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM</td>
<td>N/A</td>
<td>56.1</td>
<td></td>
</tr>
<tr>
<td>Smoothed UM</td>
<td>65.6</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Proposed method(μ=0.4)</td>
<td>83.3</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Proposed method(μ=0.5)</td>
<td>85.6</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

I adopt MRR [10] to evaluate the different approaches for bug localization. Table 1 shows the comparison results of different bug localization methods in JBoss using MRR measure. Obviously, the ranking algorithm for bug localization is better than VSM-based localization and smoothed UM-based localization which are described in S. Rao and A. Kak’s research [4].

4. Conclusion

In this study, I propose a novel method for bug triage and localization based on bug reports classification. Comparing with previous research, I adopt a smoothed UM instead of VSM to search similar bug reports when a new bug comes. Then by introducing the novel bug triage algorithm and bug localization mechanism, I can recommend the best developer and locate the correct source code files for fixing a given bug. By executing the evaluation on different methods or models, the results showed that proposed method performed better on JBoss and Eclipse.

In the future, I will use the algorithm to large-scale business projects for verifying its effectiveness. Moreover, I plan to extend the bug types for classifying more bug reports, and introduce more features (e.g., component) for conducting bug triage. For bug localization, I will extract complete source code files from other open source projects for validating the method.

References


Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(No.2012-0007149) and by Seoul Creative Human Development Program funded by Seoul Metropolitan Government (No.HM120006).