JES
Journal of Engineering Sciences

Vol 13, Issue 05, May /2022

ISSN NO:0377-9254

Mapping Bug Reports to Relevant Files: A Ranking Model, a Fine-Grained Benchmark, and Feature Evaluation

N.Harshitha, P.T.Sirisha
DEPARTMENT OF MCA
Sri Padmavathi College Of Computer Sciences & Technology

Abstract—When a new bug report is received, developers usually need to reproduce the bug and perform code reviews to find the cause, a process that can be tedious and time consuming. A tool for ranking all the source files with respect to how likely they are to contain the cause of the bug would enable developers to narrow down their search and improve productivity. This paper introduces an adaptive ranking approach that leverages project knowledge through functional decomposition of source code, API descriptions of library components, the bug-fixing history, the code change history, and the file dependency graph. Given a bug report, the ranking score of each source file is computed as a weighted combination of an array of features, where the weights are trained automatically on previously solved bug reports using a learning-to-rank technique. We evaluate the ranking system on six large scale open source Java projects, using the before-fix version of the project for every bug report. The experimental results show that the learning-to-rank approach outperforms three recent state-of-the-art methods. In particular, our method makes correct recommendations within the top 10 ranked source files for over 70 percent of the bug reports in the Eclipse Platform and Tomcat projects.

I. INTRODUCTION

A Software bug or defect is a coding mistake that may cause an unintended or unexpected behavior of the software component . Upon discovering an abnormal behavior of the software project, a developer or a user will report it in a document, called a bug report or issue report.

A bug report provides information that could help in fixing a bug, with the overall aim of improving the software quality. A large number of bug reports could be opened during the development life-cycle of a software product. For example, there were 3,389 bug reports created for the Eclipse Platform product in 2013 alone.

In a software team, bug reports are extensively used by both managers and developers in their daily development process. A developer who is assigned a bug report usually needs to reproduce the abnormal behavior [35] and perform code reviews [4] in order to find the cause.

However, the diversity and uneven quality of bug reports can make this process nontrivial. Essential information is often missing from a bug report [11]. Bacchelli and Bird [4] surveyed 165 managers and 873 programmers, and reported that finding defects requires a high level understanding of the code and familiarity with the relevant source code files.

In the survey, 798 respondents answered that it takes time to review unfamiliar files. While the number of source files in a project is usually large, the number of files that contain the bug is usually very small.

Therefore, we believe that an automatic approach that ranked the source files with respect to their relevance for the bug report could speed up the bug finding process by narrowing the search to a smaller number of possibly unfamiliar files.

II. PROJECT DESCRIPTION

OVERVIEW OF THE PROJECT

- **INSTANCE SELECTION**
- **DATA REDUCTION**
- **GRAPH MODULE**

INSTANCE SELECTION

In this module a user has to upload its files in a cloud server, he/she should register first. Then only he/she can be able to do it. For that he needs to fill the details in the registration form. These details are maintained in a database. In this module, any of the above mentioned person have to login, they should login by giving their name and password

DATA REDUCTION

To save the labor cost of developers, the data reduction for bug triage has two goals.

Reducing the data scale.

Improving the accuracy of bug triage.

In contrast to modeling the textual content of bug reports in existing work, we aim to augment the data set to build a preprocessing approach, which can be applied before an existing bug triage approach. We explain the two goals of data reduction Follows.

GRAPH MODULE

Firstly it will show how many bugs are not assigned to any developer. It will give complete status about the bugs to the admin so that he will come to know which bugs are not assigned yet.

III. SYSTEM ANALYSIS

EXISTING SYSTEM

To investigate the relationships in bug data, Sandusky et al. form a bug report network to examine the dependency among bug reports.

Besides studying relationships among bug reports, Hong et al. build a developer social network to examine the collaboration among developers based on the bug data in Mozilla project. This developer social network is helpful to understand the developer community and the project evolution.

By mapping bug priorities to developers, Xuan et al. identify the developer prioritization in open source bug repositories. The developer prioritization can distinguish developers and assist tasks in software maintenance.

To investigate the quality of bug data, Zimmermann et al. design questionnaires to developers and users in three open source projects. Based on the analysis of questionnaires, they characterize what makes a good bug report and train a classifier to identify whether the quality of a bug report should be improved.

Duplicate bug reports weaken the quality of bug data by delaying the cost of handling bugs. To detect duplicate bug reports, Wang et al. design a natural language processing approach by matching the execution information.

DISADVANTAGES OF EXISTING SYSTEM

Traditional software analysis is not completely suitable for the large-scale and complex data in software repositories.

In traditional software development, new bugs are manually triaged by an expert developer, i.e., a human triager. Due to the large number of daily bugs and the lack of expertise of all the bugs, manual bug triage is expensive in time cost and low in accuracy.

IV. PROPOSED SYSTEM

In this paper, we address the problem of data reduction for bug triage, i.e., how to reduce the bug data to save the labor cost of developers and improve the quality to facilitate the process of bug triage.

Data reduction for bug triage aims to build a small-scale and high-quality set of bug data by removing bug reports and words, which are redundant or non-informative.

In our work, we combine existing techniques of instance selection and feature selection to simultaneously reduce the bug dimension and the word dimension. The reduced bug data contain fewer bug reports and fewer words than the original bug data and provide similar information over the original bug data. We evaluate the reduced bug data according to two criteria: the scale of a data set and the accuracy of bug triage.

In this paper, we propose a predictive model to determine the order of applying instance selection and feature selection. We refer to such determination as prediction for reduction orders.

Drawn on the experiences in software metrics,1 we extract the attributes from historical bug data sets. Then, we train a binary classifier on bug data sets with extracted attributes and predict the order of applying instance selection and feature selection for a new bug data set.

ADVANTAGES OF PROPOSED SYSTEM

Experimental results show that applying the instance selection technique to the data set can reduce bug reports but the accuracy of bug triage may be decreased.

Applying the feature selection technique can reduce words in the bug data and the accuracy can be increased.

Meanwhile, combining both techniques can increase the accuracy, as well as reduce bug reports and words.

Based on the attributes from historical bug data sets, our predictive model can provide the accuracy of 71.8 percent for predicting the reduction order.

We present the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspects, namely a) to simultaneously reduce the scales of the bug dimension and the word dimension and b) to improve the accuracy of bug triage.

We propose a combination approach to addressing the problem of data reduction. This can be viewed as an application of instance selection and feature selection in bug repositories.

We build a binary classifier to predict the order of applying instance selection and feature selection. To our knowledge, the order of applying instance selection and feature selection has not been investigated in related domains.

V. SYSTEM CONFIGURATION

HARDWARE CONFIGURATION

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 1 GB.

SOFTWARE CONFIGURATION

- Operating system : Windows XP/7/8.
- Coding Language: JAVA/J2EE
- IDE : Netbeans 7.4
- Database : MYSQL

VI. CONCLUSION

Bug triage is an expensive step of software maintenance inboth labor cost and time cost. In this paper, we combine feature selection with instance selection to reduce the scale of bug data sets as well as improve the data quality.

To determine the order of applying instance selection and feature selection for a new bug data set, we extract attributes of each bug data set and train a predictive model based on historical data sets. We empirically investigate the data reduction for bug triage in bug repositories of two large open source projects, namely Eclipse and Mozilla. Our work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

In future work, we plan on improving the results of data reduction in bug triage to explore how to prepare a high quality bug data set and tackle a domain-specific software task. For predicting reduction orders, we plan to pay efforts to find out the potential relationship between the attributes of bug data sets and the reduction orders.

REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?"in Proc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361–370.

[2] S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D.Ernst, "Finding bugs in web applications using dynamic test generationand explicit-state model checking," IEEE Softw., vol. 36,no. 4, pp. 474–494, Jul./Aug. 2010.

[3] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage:Recommenders for development-oriented decisions," ACMTrans. Soft. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.

[4] C. C. Aggarwal and P. Zhao, "Towards graphical models for textprocessing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

[5] Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/

[6] K. Balog, L. Azzopardi, and M. de Rijke, "Formal models forexpert finding in enterprise corpora," in Proc. 29th Annu. Int. ACMSIGIR Conf. Res. Develop. Inform. Retrieval, Aug. 2006, pp. 43–50.

[7] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction usingquad tree-based k-means clustering algorithm," IEEE Trans.Knowl. Data Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.

[8] H. Brighton and C. Mellish, "Advances in instance selection forinstance-based learning algorithms," Data Mining Knowl. Discovery,vol. 6, no. 2, pp. 153–172, Apr. 2002.

[9] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Informationneeds in bug reports: Improving cooperation between developersand users," in Proc. ACM Conf. Comput. Supported CooperativeWork, Feb. 2010, pp. 301–310.

[10] V. Bol_on-Canedo, N. S_anchez-Maro~no, and A. Alonso-Betanzos,"A review of feature selection methods on synthetic data," Knowl.Inform. Syst., vol. 34, no. 3, pp. 483–519, 2013.

[11] V. Cerver_on and F. J. Ferri, “Another move toward the minimumconsistent subset: A tabu search approach to the condensed nearestneighbor rule,” IEEE Trans. Syst., Man, Cybern., Part B, Cybern.,vol. 31, no. 3, pp. 408–413, Jun. 2001.