ISSN:0975-9646

A Literature Survey on Efficient Software Bug Triaging Using Software Data Reduction Techniques

Prof. Subhash Pingale[#1], Prof. R.A.Taklikar[#2], Ankita Godse[#3]

[#]*ME Computer (Engineering), Dept. of Computer Science & Engineering, Solapur University, SKN Sinhgad College of Engineering, Korti, Pandharpur, Maharashtra, India*

***Abstract*— Large open source software projects receive abundant rates of submitted bug reports. Triaging these incoming reports manually is error-prone and time consuming. The goal of bug triaging is to assign potentially experienced developers to new-coming bug reports. To reduce time and cost of bug triaging, work presents an automatic approach to predict a developer with relevant experience to solve the new coming report. The proposed work combines instance selection with keyword selection to simultaneously reduce data scale on the bug dimension and the word dimension. To determine the order of applying instance selection and keyword selection, extract attributes from existing bug data sets and build a predictive model for a new bug data set. In addition, work re-balances the load between developers based on their experience; also priority level to the new bug report will be assigned with ranking to the predicted list of developers. Proposed work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.**

***Keywords*— Bug triage, data reduction, Instance selection, Keyword selection, Data Mining, Mining software repositories.**

I. Introduction

Software repositories comprise valuable information about software projects. This information can help to manage the progress of these projects. One of the important software repositories is the bug tracking system (BTS). Many open source software projects have an open bug repository that allows both developers and users to submit defects or issues in the software. BTS to manage bug reports submitted by users, testers, and developers.

Software companies spend over 45 percent of cost in fixing bugs. Due to the daily-reported bugs, a large number of new bugs are stored in bug repositories. There are two challenges related to bug data that may affect the effective use of bug repositories in software development tasks, namely the large scale and the low quality. On one hand, due to the daily-reported bugs, a large number of new bugs are stored in bug repositories. On the other hand, software techniques suffer from the low quality of bug data. Two typical characteristics of low-quality bugs are noise and redundancy. Noisy bugs may mislead related developers while redundant bugs waste the limited time of bug handling

A time-consuming step of handling software bugs is bug triage, which aims to assign a correct developer to fix a new bug. 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.

Data reduction is the transformation of numerical or alphabetical digital information derived empirically or experimentally into a corrected, ordered, and simplified form. The basic concept is the reduction of multitudinous amounts of data down to the meaningful parts.

The proposed work addresses 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. The work has the two goals of data reduction as : Reducing the Data Scale & Improving the Accuracy.

II. Literature Survey

To avoid the expensive cost of manual bug triage, an automatic bug triage approach was proposed, which applies text classification techniques to predict developers for bug reports. In this approach, a bug report is mapped to a document and a related developer is mapped to the label of the document. Then, bug triage is converted into a problem of text classification and is automatically solved with mature text classification techniques, e.g., Naive Bayes. Based on the results of text classification, a human triager assigns new bugs by incorporating his/her expertise. [2]

For a predicted list of developers by a classifier, in Developer Prioritization developers were ranked by the priorities. Thus, the developer prioritization is used to discriminate the developers with similar probabilities in the prediction. [3]

While in Profile Oriented Developer Recommendation an approach where profile is created for each developer based on his previous work. This profile is mapped to a domain mapping matrix which indicates the expertise of each developer in their corresponding area. [4]

To avoid low-quality bug reports in bug triage, a semi-supervised classifier were trained by combining unlabeled bug reports with labeled ones that improves the classification accuracy with both the labeled and unlabeled bug reports. To adjust bug triage, a weighted recommendation list (WRL) is proposed to augment the effectiveness of unlabeled bug reports. This WRL is employed to probabilistically label an unlabeled bug report with multiple relevant developers instead of a single relevant developer. [5]

For the text representation and processing a concept of distance graphs is proposed. Distance graphs represents the document in terms of the distances between the distinct words. The distance graph representation maintains information about the relative placement of words with respect to each other. Provide a much richer representation in terms of sentence structure of the underlying data. [6]

Table 1: Literature Survey Table

Paper	**Proposed**	**Conclusion**
Who should fix this bug? [2]	A semi-automated approach uses a supervised machine learning algorithm to suggest developers who may be qualified to resolve the bug.	Based on the results of text classification, a human triager assigns new bugs by incorporating his/her expertise.
Developer prioritization in bug repositories [3]	For a predicted list of developers by a classifier, developers were ranked by the priorities after bug triage.	A task-based developer prioritization were not applied in bug repositories to improve a specified task with the developer rankings.
Bug Triaging: Profile Oriented Developer Recommendation [4]	Here proposed an approach where profile is created for each developer based on his previous work and is mapped to a domain mapping matrix which indicates the expertise of each developer in their corresponding area.	It utilizes the expertise profile of developers maintained in Domain Mapping Matrix (DMM).
Towards graphical models for text processing," Knowl. Inform. Syst [6]	For the text representation and processing a concept of distance graphs is proposed.	The detail study of the problems of similarity search, plagiarism detection, and its applications wasn't specified.

To investigate the quality of bug data, questionnaires to developers and users in three open source projects was designed. 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. [10]

III. SCOPE

The scope of proposed work as follows :
1. Assigning priority levels to the new bug report.
2. Assigning ranking to the predicted list of developers.
3. Instance selection can remove uninformative bug reports
4. Keyword selection can remove uninformative words, keyword selection improves the accuracy of bug triage.

IV. PROPOSED SYSTEM

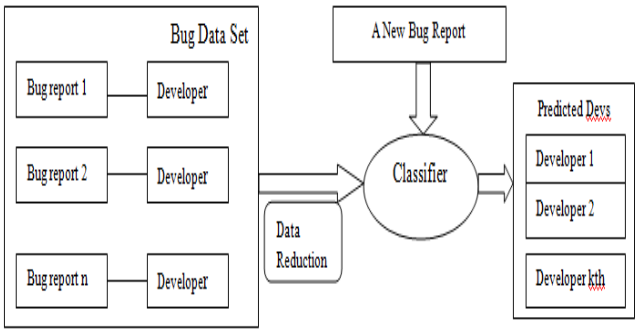

Fig 1: System Architecture

Fig. 1 illustrates the basic framework of bug triage based on text classification. A bug data set contains bug reports with respective developers. On this bug data set the bug data reduction is applied as a phase in data preparation of bug triage. Work combines existing techniques of instance selection and keyword selection to remove certain bug reports and words. A problem for reducing the bug data is to determine the order of applying instance selection and keyword selection, which is denoted as the prediction of reduction orders.

A. Applying Instance Selection and Keyword Selection:

In bug triage, a bug data set is converted into a text matrix with two dimensions, namely the bug dimension and the word dimension. Work leverages the combination of instance selection and keyword selection to generate a reduced bug data set. Replace the original data set with the reduced data set for bug triage. For a given data set in a certain application, instance selection is to obtain a subset of relevant instances (i.e., bug reports in bug data) while keyword selection aims to obtain a subset of relevant words in bug data. To distinguish the orders of applying instance selection and keyword selection, we give the following denotation. Given an instance selection algorithm IS and a keyword selection algorithm KS, we use KS!IS to denote the bug data reduction, which first applies KS and then IS; on the other hand, IS!KS denotes first applying IS and then KS. In Algorithm 1, we briefly present how to reduce the bug data based on KS !IS. Given a bug data set, the output of bug data reduction is a new and reduced data set. Two algorithms KS and IS are applied sequentially.

B. Reduction Orders :

To apply the data reduction to each new bug data set, need to check the accuracy of both two orders (KS!IS and IS!KS) and choose a better one. To avoid the time cost of manually checking both reduction orders, consider predicting the reduction order for a new bug data set based on historical data sets.

V. CONCLUSIONS

The proposed work presents an approach to automatically assign bug reports to developers with the appropriate expertise. The proposed work combines keyword selection with instance selection to reduce the scale of bug data sets as well as improves the data quality. This work provides an approach for leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

REFERENCES

[1] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Towards Effective Bug Triage with Software Data Reduction Techniques," IEEE Transactions on Knowledge & Data Engineering, Vol. 27, No. 1, JAN 2015.

[2] 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.

[3] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in Proc. 34th Int. Conf. Softw. Eng.(ICSE), 2012, pp. 25–35.

[4] Anjali Sandeep Kumar Singh, "Bug Triaging: Profile Oriented Developer Recommendation," (IJIRAE) ISSN: 2349-2163 Volume 2 Issue 1 (January 2015).

[5] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Automatic bug triage using semi-supervised text classification," in Proc. 22nd Int. Conf. Softw. Eng. Knowl. Eng., Jul. 2010, pp. 209–214.

[6] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, July 2012.

[7] G. Canfora and L. Cerulo. How software repositories can help in resolving a new change request, in Workshop on Empirical Studies in Reverse Engineering, 2005.

[8] J. Anvik, "Automating bug report assignment," in Proc 28th International Conference on Software Engineering. ACM, 2006, pp. 937–940.

[9] W. Zou, Y. Hu, J. Xuan, and H. Jiang, "Towards training set reduction for bug triage," in Proc. IEEE 35th Annual Computer Software and Applications Conference, Washington, DC, USA: IEEE Computer Society, 2011, pp. 576–581.

[10] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 618–643, Oct. 2010.