International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

A Survey on Bug Triaging- Software Data Reduction Techniques

Vijay Kukre[1], Shyam Gupta[2]

[1]PG Student, Siddhant College of Engineering, Sudumbre, Savitribai Phule Pune University

[2]Professor, Computer Department, Siddhant College of Engineering, Sudumbre, Savitribai Phule Pune University

Abstract: ***Most of the package corporations have to modify sizable amount of package bugs a day. Package bugs square measure inescapable and fixing package bugs is a rich task. The goal of effective bug triaging package is to assign doubtless intimate developers to new-coming bug reports to cut back time and price of bug triaging, An automatic approach is planned during this paper that predicts a developer with relevant expertise to resolve or fix the new returning bug report during this paper, the five term choice strategies on the accuracy of bug assignment square measure used. Additionally, the load between developer supported their expertise is re-balanced. The planned system is made with intention to counsel or advocate the bug and to not mechanically assign it. this enables a window to handle real time crisis that come back up throughout project development lifecycle.***

Keywords: Mining software repositories, application of data pre-processing, data management in bug repositories, bug data reduction, feature selection, instance selection, bug triage.

1. Introduction

Many software package corporations pay most of the money in fixing the bugs. Massive software package comes have bug repository that collects all the knowledge associated with bugs. In bug depository, every software package bug incorporates a bug report. The bug report consists of matter info concerning the bug and updates associated with standing of bug fixing. Once a bug report is made, a personality's triager assigns this bug to a developer; World Health Organization can try and fix this bug. This developer is recorded in associate item assigned-to. The assigned to will amendment to a different developer if the antecedently assigned developer cannot fix this bug. the method of assigning an accurate developer for fixing the bug is termed bug triage. Bug sorting is one among the foremost time intense step in handling of bugs in software package comes. Manual bug sorting by a personality's triager is time intense and erring since the quantity of daily bugs is massive and lack of information in developers regarding all bugs. Because of all these things, bug sorting ends up in costly time loss, high price and low accuracy. The information keep in bug reports has 2 main challenges. First of all the massive scale information and second low quality of knowledge as a result of sizable amount of daily reported bugs, the number of bug reports is scaling up within the repository. Noisy and redundant bug's square measures degrading the standard of bug reports. In this paper an efficient bug sorting system is projected which scale will back the bug information to save lots of the labor price of developers. It conjointly aims to create a top quality set of bug data by removing the redundant and non-informative bug reports.

2. Literature Survey

In [1] they mention that Bug triaging is a fallible, tedious and time intense task. They're going with Revisiting Bug sorting and determination Practices. In this paper they studied regarding bug triaging and fixing practices, including bug reassignments and re-openings, within the context of the Mozilla Core and Firefox comes, which they consider to be representative samples of a large-scale open source computer code project. Additionally they need conceive to conduct qualitative and qualitative analysis of the bug assignment practices. They have a tendency to have an interest in providing insights into several areas: sorting practices, review and approval processes; root cause analysis of bug reassignments and reopens in open supply computer code projects; and recommendations for improvements/ redesign of bug tracking systems.

In [2] this paper, they introduce a graph model supported Markov chains that captures bug moving history. This model has many fascinating qualities. First, it reveals developer networks which might be wont to discover team structures and to search out appropriate consultants for a replacement task. Second, it helps to higher assign developers to bug reports. In our experiments with 445,000 bug reports, our model reduced moving events, by up to seventy two additionally, the model exaggerated the prediction accuracy by up to twenty three percentage points compared to ancient bug triaging approaches.

In [3] recent analysis shows that optimizing recommendation accuracy drawback and proposes an answer that is basically an instance of content-based recommendation (CBR). However, CBR is well known to cause over-specialization, recommending solely the categories of bugs that every developer has solved before. This drawback is critical in apply, as some veteran developers may be over laden, and this is able to slow the bug fixing method. In this paper, they take 2 directions to handle this problem: initial, we have a tendency to develop the matter as an optimization drawback of each accuracy and price. Second, we adopt a content-boosted cooperative filtering (CBCF), combining an

existing CBR with a cooperative filtering recommender (CF), which boosts the advice quality of either approach alone.

In [4] Current techniques either use data retrieval and machine learning to search out the foremost similar bugs already fixed and suggest knowledgeable developers, or they analyze change data stemming from ASCII text file to propose expert bug solvers. Neither technique combines matter similarity with modification set analysis nor thereby do exploits the potential of the complex between bug reports and alter Levant options (i.e., words in bug data) within the planned system, the mixture of instance choice and have selection is employed. The planned systems are enforced in java language thus it'll be platform freelance. As there's no restriction on the dimensions of bug's info, a tester will add large number of bugs within the system. this can be one in all the biggest benefits of the planned system. Since all the bug's info is receptive all the developers, it takes less time for the developer to require the choice. Developer will quickly opt for the bug to repair. Since bug sorting aims to predict the developers who will fix the bugs, we have a tendency to follow the present work to get rid of unfixed bug reports, e.g., the new bug reports or will-not-fix bug reports. Thus, we have a tendency to solely opt for bug reports, that area unit mounted and duplicate (based on the things standing of bug reports). Moreover, in bug repositories, many developers have solely fixed only a few bugs. Such inactive developers might not provide enough info for predicting correct developers. In our work, we have a tendency to take away the developers, who have mounted but ten bugs.

In [5], they suggest a semi-supervised text classification approach for bug triage to avoid the insufficiency of labeled bug reports in existing supervised approaches which combines naive Bayes classifier and expectation maximization to take benefit of both labeled and unlabeled bug reports. This approach trains a classifier with a part of labeled bug reports and then the approach iteratively labels many unlabeled bug reports and trains a new classifier with labels of all the bug reports. They also used a weighted advice list to improve the performance by daunting the weights of multiple developers in training the classifier. Experimental outcome on bug reports of Eclipse demonstrate that new approach is good than the existing supervised approaches in terms of classification accuracy of bug triage by up to 6% but does not provide automatic bug triage with a bug repository.

In [6], they examine the make use of five term selection methods on the correctness of bug task to minimize time and cost of bug triaging and also re-balance the load between developers based on their knowledge so, they carry out experiments on four real datasets. The first term selection method, Log Odds Ratio (LOR) measures the odds of the word occurring in the positive class normalized by the negative class. The second term selection method, Chi-Square (X2) test is used to observe independence of two events. The third term selection method, Term Frequency Relevance Frequency (TFRF) is used to select more high frequency for instances in the positive category than in the negative category. The fourth term selection method, Mutual Information (MI) is used to measures the shared dependence of two random variables. The fifth term selection method, Distinguishing Feature Selector (DFS) gives total prejudiced powers of the features over the entire text set rather than being class specific. The investigational outcome demonstrates that by selecting a small number of discriminating terms, the F-score can be significantly enhanced.

In [7], they propose the combination of both feature selection and instance selection techniques to improve the accuracy of bug triage to evaluate the training set reduction on the bug data of Eclipse. As a result, 70% words and 50% bug reports are removed after the training set reduction. The experimental results show that the new and small training sets can provide better accuracy than the original one. The drawbacks of their approach are low precision rate and cannot be directly transferred to other projects as the results are based on parts of the bug data from the Eclipse only.

In [8], they applied conventional bug triage techniques to projects of different sizes and found that the usefulness of a bug triage technique mainly depends on the size of a project team i.e. the number of developers, hence become less useful when the number of developers increases. They proposed a method called BugFixer shown in Figure 1, makes a new bug report based on historical bug-fix information and constructs a Developer-Component-Bug (DCB) network, which shows the association between developers and source code components and also the association between the components and their related bugs. A DCB network finds the information such as "who fixed what, where". For a new bug report, BugFixer uses a DCB network to suggest to triager a list of appropriate developers who might fix this bug. The experimental outcome of their methods outperforms the existing methods for large projects and achieves as good as performance for small projects.

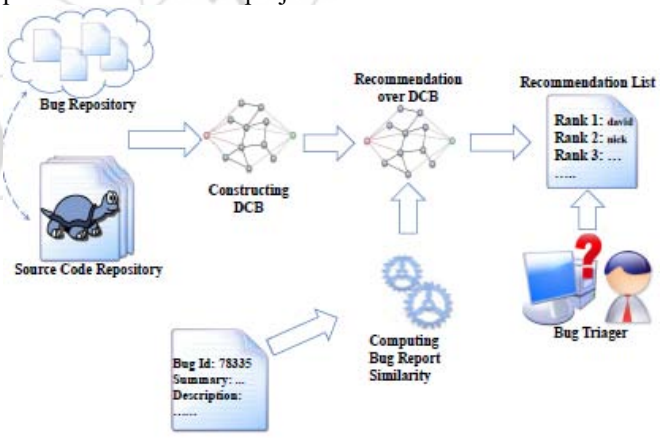

Figure 1: Structure of BugFixer

3. Proposed System

The diagram in figure 2 illustrated the system architecture of the proposed system. The input to the system is in the form of bug data set. The bug data set consists all the details of software bugs. Each bug has bug report and the details of the developer who have worked on that respective bug. The bug

report is mainly divided in two parts, summary and description. The proposed system gives predicted results in form of output. Basically, there are two types of users in the proposed system. First is the developer and second is the tester. Developer will get software bugs assigned to him. Developer can work on only one software bug at a time. Tester can add new bugs to the system. As shown in figure 2, the proposed system makes use of bug data reduction. In the proposed system, to save the labor cost of developers, the data reduction for bug triage is made. Bug data reduction is applied in phase of data preparation of bug triage.

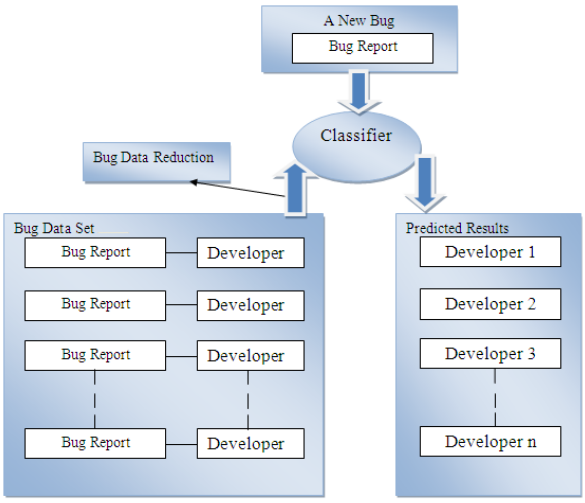

Figure 2: System Architecture

Data reduction mainly has two goals. Firstly, reducing the data scale and secondly, improving the accuracy of bug triage. Techniques of instance selection and feature selection are used for data reduction. Instance selection and feature selection are widely used techniques in data processing. 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 feature selection aims to obtain a subset of relevant features (i.e., words in bug data). In the proposed system, the combination of instance selection and feature selection is used.

4. Conclusion

Bug triage is a chip step of computer code maintaining. The projected system aims to form reduced and high-quality bug knowledge in computer code development and maintenance. Processing techniques like instance choice and have choice are used for data reduction. The projected system is used for any open supply comes that generate immense bug knowledge. Various software corporations engaged on comes like banking, food chain management will use the applying of the projected system. The advantage of proposed system is, it combines feature selection with instance selection to decrease the level of bug data sets as well as improve the data quality. The next advantage is, it provide priority according to severity of bug and security so that no another developer can access it.

References

[1] Baysal, O., Holmes, R., & Godfrey, M. W. (2012, June), "Revisiting bug triage and resolution practice" In *User Evaluation for Software Engineering Researchers (USER), 2012* (pp. 29-30) IEEE.

[2] Jeong, G., Kim, S., & Zimmermann, T. (2009, August), "Improving bug triage with bug tossing graphs" in *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering* (pp. 111-120). ACM.

[3] Park, Jin-woo, Mu-Woong Lee, Jinhan Kim, Seung-won Hwang, and Sunghun Kim, "CosTriage: A Cost-Aware Triage Algorithm for Bug Reporting Systems." In *AAAI*. 2011..

[4] Kevic, Katja, Sven Christian Muller, Thomas Fritz, and Harald C. Gall. "Collaborative bug triaging using textual similarities and change set analysis", In *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on*, pp. 17-24. IEEE, 2013..

[5] Xuan, Jifeng, He Jiang, Zhilei Ren, Jun Yan, and Zhongxuan Luo "Automatic Bug Triage using Semi-Supervised Text Classification" in *SEKE*, pp. 209-214, 2010..

[6] Alenezi, Mamdouh, Kenneth Magel, and Shadi Banitaan. "Efficient bug triaging using text mining." *Journal of Software* 8.9 (2013): 2185-2190.

[7] Zou, Weiqin, Yan Hu, Jifeng Xuan, and He Jiang. "Towards training set reduction for bug triage." In *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*, pp. 576-581. IEEE, 2011.

[8] S Hu, Hao, Hongyu Zhang, Jifeng Xuan, and Weigang Sun. "Effective bug triage based on historical bug-fix information." In *Software Reliability Engineering (ISSRE), 2014 IEEE 25th International Symposium on*, pp. 122-132 IEEE, 2014.