AN EFFICIENT APPROACH TOWARDS AUTOMATED BUGS TRIAGE SYSTEM WITH IMPLEMENTATION OF RANDOM FOREST

Sankalp. S.Wankar , Dr.Sanjeev Shrivastava,
Mtech Student Professor
Computer Science & Engineering ,
Guru Nanak Institute of Engineering and Technology ,Nagpur Kalmeshwar Road, India

Abstract : Nowadays IT companies is spending more than 40 percent of their cost in fixing software bugs, traditionally these bugs are fixed by manual assignment to a particular developer , this approach causes too much dependency, the new and alternative approach is the bug triage system which fix the bug automatically , which automatically assign the reported bug to a developer which decreases the time and cost in in manual work, different classification techniques are used to conduct automatic bug triage. In this paper, we propose to apply machine learning techniques to assist in bug triage to predict which developer should be assigned on the bug based on its description by applying text categorization. We will address the problem of data reduction for bug triage, i.e. how the quality of bug data would be improved.

IndexTerms **- Related Work, Proposed System.**

I. INTRODUCTION

Automated Bug tracking system has its significance in large software development projects which manages bug reports and list of developers who work on fixing them. Bug tracking systems has its importance in open source software development, where the team members can be dispersed around the world. In such distributed projects, the developers and other contributors may rarely see each other. Secondly, the bug tracking system is used not only to keep track of problem reports and feature requests, but it also help in coordinating the work among the different developers Automated Bug tracking system has its significance in large software development projects which manages bug reports and list of developers who work on fixing them. Bug tracking systems has its importance in open source software development, where the team members can be dispersed around the world. In such distributed projects, the developers and other contributors may rarely see each other. Secondly, the bug tracking system is used not only to keep track of problem reports and feature requests, but it also help in coordinating the work among the different developers. Software bugs can never be avoidable and at the same time fixing bugs is expensive in software development. IT companies spend more than 40 percent of their cost in fixing software bugs. Large software projects deploy bug repositories to support information collection and to assist developers to handle bugs. A bug repository will contain all report which plays an important role in managing software bugs. In a bug repository, a bug is maintained as a bug report, which records the complete description of reproducing the bug and updates according to the status of bug fixing. A bug repository provides a data platform to support many types of tasks on bugs, e.g., fault prediction, bug localization, and reopened bug analysis. The bug reports in a bug repository are called bug data. This bug report is then assigns to a developer, who starts to fix the bug. If the assigned developer is unable to fix the bug, the bug is migrated to another developer. The process of assigning a bug report to an appropriate developer is called bug triage. As the Work of Bug triage system is to choose the appropriate developer for fixing bugs, we will follow the existing work to remove unfixed bug reports.

II. RELATED WORK

Cubrani and Murphy First proposed the problem of automatic bug triage. The Machine Learning technique, Text Categorization is applied to assist in bug triage by using text categorization [1]. Text categorization is also known as text classification which is a technique of automatically sorting a set of documents into categories from a predefined set where a

developer gets predicted using the bug's description [2]. For this they used supervised machine learning technique using Naïve Bayes classifier to predict the correct developer. *Xuan* present a semi-supervised approach for automatic bug triage using text classification [3]. Their approach combined the naive Bayes classification approach and expectation maximization to take the advantage of both labelled and unlabelled bug reports. Xuan trains a classifier with a fraction of labelled bug reports. This approach labelled numerous unlabelled bug reports. From the result of, this semi-supervised approach improves the classification accuracy of bug triage by up to 6% and it avoids low-quality bugs. When a bug report has been assigned to a specific developer, then if the assigned developer is unable to fix the bug, the assigned developers can forward or reassign the bug to other developer. This process of reassignment of bug from one developer to another is called "Bug Tossing". Jeong find out that in manual bug triage, 37 percent - 44 percent of bug reports are "tossed"[4].In addition, Jeong et al. a model of bug tossing is built to reduces the number of reassignment of bug reports. The Markov chains based tossing graph approach is proposed to capture the past bug tossing history which improved the bug assignment and reduced unnecessary tossing steps. P. Bhattacharya proposed a method for bug. triage [12]. Goal of His Proposed system was to find the optimal set of machine learning techniques to improve bug assignment accuracy in large projects. [5] Used a set of machine learning tools and a probabilistic graph- based model (bug tossing graphs) that lead to highly- accurate predictions, and laid the foundation for the next generation of machine learning-based bug assignment. They used methodology like Choosing effective classifiers and features, Incremental learning, Multifeatured tossing graphs to achieve their goal. The current technique of Bug Triaging involves modelling the reassignment of bugs as a goal-oriented path model [6]. V. Akila. Proposed a new framework with the additional capabilities. This models the reassignment of bugs as Enriched Adaptive Bug Triaging System (EABTS) which is based on actual path model. Their graph structure captures the relationship among developers as the number of tosses and also captures the propinquity exists among developers. Therefore, this graph structure is enriched. The technique was based on Ant routing. Ant routing is inherently adaptive in nature. Their work gave a sub graph that consists of developers who are frequently involved in bug resolution.

III. PROPOSED SYSTEM

We propose to apply machine learning techniques to assist in bug triage to predict which developer should be assigned on the bug based on its description by applying text categorization, i.e. how the quality of bug data would be improved. Following are the different scenarios for Our Proposed Bug Tracking System.

Scenario 1: In first scenario the assigned developer will resolved the report and it will be labelled by the developer's class regardless of who has submit the report or the type of resolution.

Scenario 2: In second scenario, the report can be resolved by someone other than the assigned developer, but not by the person who submitted it or nor by the person directly assigned, the report will be labelled with the class of the particular developer who marked it resolved. The reasoning is that whoever resolve the report is the person to whom it should have been assigned.

Scenario 3: In this scenario, if the report is resolved as fixed, Regardless of who the resolver was, we assume that this is the developer who implemented the fix and label the report with the class of the assigned developer, as He is the person who had done the real work on the report. This rule covers the frequent case where an Eclipse developer files a report, which is then assigned to somebody else or a sub-team alias by default, and then later implements the fix himself.

Scenario 4: If the report was resolved as non-fixed by the person who submitted it, and who was not also assigned to it, the report is class labelled with the first assigned developer or person. Since it might be a feature or a bug which will later be handled by someone or after being prompted by a developer for details of his or her setup and discover that there is no bug.

Scenario 5: If the submitter resolve the report as non- fixed who was not the assigned-to developer, and nobody responded, there is an error or the submitter caught the mistake before anyone started to solve the issue, these reports are removed from the training set, as it cannot be reliably labelled.

Scenario 6: If no developer could resolve the report then it is marked as non-fixed and the class is assigned by the developer who was the last person who has worked on the report.

IV. ACKNOWLEDGMENT

Bug triage is one of the expensive step in software maintenance for both cost as well as time. Our Approach will 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 will extract the attributes of each bug dataset and will train a predictive model based on historical bug datasets. We will empirically investigate the data reduction for bug triage in bug repositories of two large open source projects, namely Eclipse and Mozilla. Our Proposed system will be based on Random Forest which will provide an efficient approach on data processing to reduce the scale and provide high-quality bug data in software development and maintenance.

REFERENCES

1. D. Cubrani and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Know l. Eng., Jun. 2004, pp. 92–97.
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, 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
4. G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with tossing graphs," in Proc. Joint Meeting 12th Eur. Softw. Eng. Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng., Aug. 2009, pp. 111–120.
5. T. Zhang, G. Yang, B. Lee, I. Shin "Role Analysis-based Automatic Bug Triage Algorithm", 2012
6. V.Akila, Dr.G.Zayaraz, Dr.V.Govindasamy "Effective Bug Triage – A Framework", International Conference on Intelligent Computing, Communication & Convergence, 114 – 120, 2015.
7. 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 generation and explicit-state model checking," IEEE Softw., vol. 36, no. 4, pp. 474–494, Jul./Aug. 2010
8. J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," ACM Trans. Soft. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011
9. C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013
10. Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/
11. K. Balog, L. Azzopardi, and M. de Rijke, "Formal models for expert finding in enterprise corpora," in Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, Aug. 2006, pp. 43–50.
12. P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Trans. Knowl. Data Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.
13. H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," Data Mining Knowl. Discovery, vol. 6, no. 2, pp. 153–172, Apr. 2002.
14. S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in Proc. ACM Conf. Comput. Supported Cooperative Work, Feb. 2010, pp. 301–310.
15. Bolon-Canedo, N. S anchez-Marono, and Alonso-Betanzos, "A review of feature selection methods on synthetic data," Knowl. Inform. Syst., vol. 34, no. 3, pp. 483–519, 2013.