A Survey on Bug Triage Using Data Reduction Technique

Swati Bhaskar Ghusalkar
Student
Dept. of Computer Engg.
DGOI, COE, Daund, Pune
s.ghusalkar@gmail.com

Prof. Sachin Bere
Assistant Professor
Dept. of Computer Engg
DGOI, COE, Daund, Pune
sachinbere@gmail.com

Abstract **-**Most of the software companies needs to deal with software bug in every day. Software companies spend most if their cost in dealing with software bugs. The process of fixing bug is bug triage, which aims to assign a expert developer to a new bug. To reduce the time and cost in manual work, we apply text classification technique to conduct automatic bug triage. In proposed system we apply data reduction techniques on bug data set to improve the scale and quality of bug data. We use instance selection and feature selection simultaneously to reduce the scales on bug dimension and word dimension and improve the accuracy of bug triage. In this paper, we investigate the use of five term selection methods on the accuracy of bug assignment. In addition, we re-balance the load between developers based on their experience.

Keywords*- Bug, Bug triage, Data Management in Bug Repositories, Data Reduction, Instance Selection, Feature Selection*

*****

I. INTRODUCTION

Most of the software companies need to deal with large number of software bugs every day. Software bugs are unavoidable and fixing software bugs is an expensive task. A bug repository plays an important role in managing software bugs. Software companies spend most if their cost in dealing with software bugs. In software development, software repositories are large-scale databases for storing the output of software development, e.g., source code, emails, specifications and bugs. Traditional software analysis is not completely suitable for the large-scale and complex data in software repositories. In a bug repository bug is maintained as a bug report. Bug report includes all the textual description regarding bugs and update status of bug fixing. The information stored in bug reports has two main challenges. Firstly the large scale data and secondly low quality of data. Due to large number of daily bugs reported, the number of bug reports is scaling up in the repository. Noisy and redundant bugs are degrading the quality of bug reports. A large number of daily reported bugs are stored in bug repositories. The process of fixing bug is bug triage, which aims to assign a expert developer to a new bug. In traditional software new bugs are manually triaged by expert developer i.e. human triager.

Due to the large number of daily bugs arrives in a bug repositories and the lack of expertise of all the bugs, manual bug triage is very time consuming and cost and low in accuracy. To avoid the expensive cost of manual bug we propose automatic bug triage approach which applies text classification technique to predict developer for Bug report, based on the results the expert developer assigns new bugs by his/her expertise.

In proposed system we use an effective bug triage system which will reduce the bug data to save the labor cost of developers and improve the quality to facilitate the process of bug triage.

II. DATA REDUCTION FOR 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 unnecessary. In this paper we are using feature selection and instance selection with historical data for reducing the bug data in bug repository so that we get quality data as well as low scale data.

1) APPLYING INSTANCE SELECTION AND FEATURE SELECTION

In bug triage, a bug data set is converted into a text matrix with two dimensions; they are the bug dimension and the word dimension. In this paper, we use the combination of instance selection and feature selection to generate a reduced bug data set. We change the original data set with the reduced data set for bug triage.

Instance selection and feature selection are widely used techniques in data processing. In data reduction for a given data set in a certain application, instance selection is to obtain a subset of relevant instances while feature selection is to obtain a subset of relevant features [1].

2) BENEFITS OF DATA REDUCTION

a) Reducing the data scale: In word dimension we use feature selection to remove noisy or duplicate words in a data set. Based on feature selection method, the reduced data set can be handled more easily by automatic techniques (e.g., bug triage approaches) than the original data set. In bug triage, the reduced data set can be further used for other software tasks after bug triage (e.g., severity identification, time prediction, and reopened bug analysis)

b) Improving Accuracy: In Word dimension by removing meaningless words, feature selection improves the accuracy of bug triage .This can recover the accuracy loss by instance selection.

III. ARCHITECTURAL DESCRIPTION

The system architecture of the proposed system is shown in fig. (1). In proposed system the input is in the form of bug data set. The bug data set consists of bug report and the details of the developer who have worked on that respective bug. The bug report is mainly separated in two parts, summary and description. The proposed system gives predicted results in the form of output. Basically, there are two types of users in the proposed system. First is the developer and second is the

tester. Tester can adds new arriving bug in the bug repositories. Then developer will get the system bug assigned to him by his/her expertise.

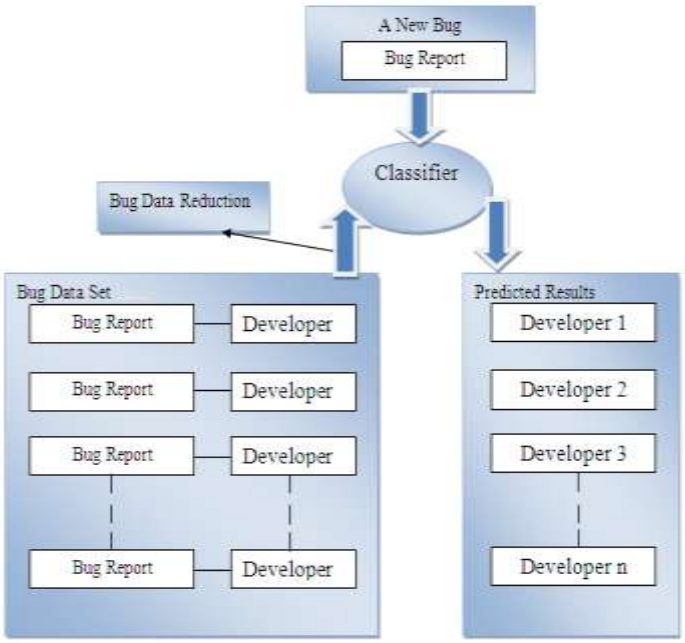

Fig 1 : Architecture of Proposed System

This system maintains the products, Bugs and bug tracking. It has advantage of maintaining bug history it stores all the details from bug origin to bug resolution. Our System provides the searching based on status, priority, and operating system

The proposed system makes use of bug data reduction. In proposed system we apply data reduction techniques on bug data set to improve the scale and quality of bug data. We use instance selection and feature selection simultaneously to reduce the scales on bug dimension and word dimension and improve the accuracy of bug triage. In bug dimension we use instance selection to reduce noisy or duplicate bug reports to decrease the number of historical bugs. And in word dimension we use feature selection to reduce noisy or duplicate words. In the system bug triage aims to predict the correct developer who can fix the bugs. We follow existing work of developer to solve the problem of fixing bugs. We predict a new bug to the developer by his/her expertise. In system bug repositories, several developers only fixed very few bugs. Such inactive developer does not provide sufficient information for assigning correct developers. In proposed system we eliminate the developers, who have fixed less than 10 bugs.

IV. REVIEW OF BUG TRIAGE

In [2] In this paper, we present a semi-automated approach propose to simplicity one part of this process, the task of reports to a developer. Our approach uses a machine learning algorithm to the open bug repository to study the kinds of reports each developer resolves. When a new report arrives, the machine learning technique suggests developers suitable to resolve the bug.

In [3] author presents a dynamic test generation technique for the dynamic Web applications. The technique uses both combined concrete and symbolic execution and explicit-state model checking. This technique generates tests automatically, by capturing logical constraints on inputs it runs the tests, and minimizes the conditions on the inputs so that resulting bug reports are small and useful in finding and fixing the underlying faults.

In [4] To better understand the relationship between developer and user we have quantitatively and qualitatively examine the questions from the MOZILLA and ECLIPSE projects. We categorized the questions and examine response and time by category and project. In this paper results show that the role of users goes more than simply reporting bugs: their active and ongoing participation is important for making progress on the bugs they report.

In [5] author propose a graph based model Markov chains, which captures bug tossing history. This model has several pleasing qualities. First, it displays developer networks which can be used to find out team structures and to find suitable experts for a new task. Second, it helps to assign expert developers to bug reports. In our experiments with 445,000 bug reports, our model reduced tossing events, by up to 72%. In addition, the model increased the prediction accuracy by up to 23 percentage points compared to traditional bug triaging approaches.

In [6] author propose to apply machine learning techniques to assist in bug triage by using text categorization to predict the developer that should work on the bug based on the bug's description. Our approach demonstrates on a collection of 15,859 bug reports from a large open-source project. Our evaluation shows that using supervised Bayesian learning, can correctly predict 30% of the report assignments to developers.

In [8] author deals with the bug tracking system, different testers submit multiple reports on the same bugs that they encounter, referred to as duplicates, which may cost more efforts in triaging and fixing bugs. In order to detect duplicates accurately, we propose a retrieval function (REP) to measure the similarity between two bug reports. It uses the information available in a bug report including the similarity of textual content in summary and description fields, also similarity of non-textual fields such as product, version, and component.

In [9] author surveyed the problem of the developer prioritization, which ranks the contributions of developers. We mainly discover two aspects, namely modeling the developer prioritization in a bug repository and assisting predictive tasks with our model. First based on social network technique, we model how to assign the priorities of developers. Second, we consider the developer prioritization to improve bug repositories.

In [11] the goal of bug triage is to assign correct developer to new coming bugs in bug data set. The bug triage approach based on the machine learning algorithm, which build classifiers from the training bug data set. In this paper we

propose data reduction techniques using instance selection and feature selection for bug triage.

In [14] author surveyed that software systems the number of daily reported bugs is high. Triaging these bugs is time consuming process. Bug triage is the process of fixing bugs to the appropriate developer. We present an approach to automatically suggest developer who resolves that bug expertly.

In [15] when a new bug report arrives in bug repositories then triager examines whether it is duplicate or not by using natural language information to detect duplicate bug reports.

V. CONCLUSION

Software companies spend most if their cost in dealing with software bugs. Bug-Tracking System is an ideal solution to track the bugs of a product, solution or an application. The proposed system aims to form reduced and high-quality bug data in software development and maintenance. In this paper, we have presented an approach to automatically assign bug reports to developers with the appropriate expertise. Our approach used the term instance selection and feature selection methods for data reduction. Our experimental results showed that this data reduction technique will give quality data as well as it will reduce the data scale.

REFERENCES

[1] Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, and Xindong Wu," Towards Effective Bug Triage with Software Data Reduction Techniques" IEEE transactions on knowledge and data engineering, vol. 27, no. 1, January 2015

[2] John Anvik, Lyndon Hiew and Gail C. Murphy "Who Should Fix This Bug?" Department of Computer Science University of British Columbia @cs.ubc.ca

[3] 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.

[4] 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.

[5] 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.

[6] D. _Cubrani_c and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.

[7] J. W. Park, M. W. Lee, J. Kim, S. W. Hwang, and S. Kim, "Costriage: A cost-aware triage algorithm for bug reporting systems," in Proc. 25th Conf. Artif. Intell., Aug. 2011, pp. 139–144

[8] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in Proc. 26th IEEE/ACM Int. Conf. Automated Softw. Eng., 2011, pp. 253–262.

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

[10] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," in Proc. 7th IEEE Working Conf. Mining Softw. Repositories, May 2010, pp. 1–10.

[11] W. Zou, Y. Hu, J. Xuan, and H. Jiang, "Towards training set reduction for bug triage," in Proc. 35th Annu. IEEE Int. Comput. Soft. Appl. Conf., Jul. 2011, pp. 576–581.

[12] E. Murphy-Hill, T. Zimmermann, C. Bird, and N. Nagappan, "The design of bug fixes," in Proc. Int. Conf. Softw. Eng., 2013, pp. 332–341.

[13] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," in Proc. 7th IEEE Working Conf. Mining Softw. Repositories, May 2010, pp. 1–10.

[14] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug report susing a vocabulary-based expertise model of developers," in Proc.6th Int. Working Conf. Mining Softw. Repositories, May 2009,pp. 131–140.

[15] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in Proc. 30th Int. Conf. Softw. Eng., May 2008, pp. 461–470.

[16] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 249–283, 2013.