A Survey on Automatic Bug Triage Using Data Mining Concepts

K. Reshma Revathi[1], Dr. S. Kirubakaran[2]

[1]P.G Scholar Department of Computer Science and Engineering, Anna University, India

[2]Professor, Department of Computer Science and Engineering, Anna University, India

Abstract: ***In bug triage process, assigning a correct developer to fix the new incoming bug is tedious than fixing that bug. Since the number of daily new coming bug is high, manual triaging increases the development cost and time. In order to automate the bug triage process, data mining techniques such as text classifications are used. Data reduction is one of the major problem identified in applying text classification to automate the bug triage process, which decreases the accuracy of bug triaging. This paper provides thecomplete survey of automatic bug triage using data mining concept. Also, a framework called Auto-BugTriager that eliminates the problem of data reduction to greater extent in bug triaging has been proposed. The proposed framework consist of three phases namely InfoZie, DataReduction and NBPredictor which process together to predict the recommendation list of expert developer for fixing the bug. This framework focus on domain specific environment in order to extend the idea for practical use. This system improves the accuracy of bug triaging and also, the quality of development and maintenance process in software industries.***

Keywords: Bug Triage Process, Manual Triaging, Data Mining Techniques, Text Classification, Data Reduction, Domain Specific Environment.

1. Introduction

Almost, every software companies deals with the flow of bugs in all kinds of projects. In open source environment, almost 300 bugs appears every day. Where in domain specific environment, almost 30 to 100 bugs are encountered based on the size of the projects. This is much for programmers to handle by themselves since many bug reports are invalid or duplicate of another bug reports. Therefore, every bug report needs to be triaged, in order to check its validity and duplicity. One of the time-consuming step in processing bug triage is to assign an appropriate developer to fix a new bug.

In traditional software development, triaging is done manually by an expert developer, i.e., a human triage. Due to the large number of daily bugs and the lack of expertise of all the bugs, manual bug triage is expensive both in time and cost. In human triage whenever a new bug appears, an expert assigns the new bug to a developer, who will try to fix this bug. Most of the time the prediction of expert is not accurate. Manual triage is error-prone due to the large number of daily bugs and the lack of knowledge about all bugs by the developers. Human bug triage results in expensive time loss, high cost and low accuracy. Reassigning the new bug for different developers to fix lasts for months while fixing that bug takes only 2 to three days.Therefore, it is important to automate the bug triage process in every software companies for improving their production quality. Research trend employs data mining concepts to deal with the software engineering problems. Many mining concepts such as text classification, fuzzy logic, text mining, extraction methods, etc., are being applied to automate the bug triage process. Bug reports are free-form of data, which has two main challenges. First challenge is the duplicate reports available in the bug repository. Mining large scale data will only results in low accuracy. The second challenge is the low quality of data i.e., the uninformative stop words in the bug report. Both these challenges are together are called as data reduction problem, which degrades the accuracy of bug triage.

In this paper, a framework called Auto-BugTriager is proposed which aims to eliminates the problem of data reduction to greater extent in bug triaging. Our framework consist of three phases namely InfoZie, DataReduction and NBPredictor which works together to predict the recommendation list of expert developer for fixing the bug. In order to extend our idea for practical use, we propose to implement this framework in a domain specific environment. We strongly believe that this system helps in improving the accuracy of bug triaging and also, the quality of development and maintenance process in software industries.

2. Literature Review

To avoid the expensive cost of manual bug triage, automatic bug triage approaches are being proposed using data mining techniques such as text classification. Jifeng Xuan et al. [1] address the problem of data reduction for bug triage, i.e., how to reduce the bug dimension and the word dimension. They proposed to combine feature selection with instance selection algorithms to reduce the scale of bug reports as well as to improve the data quality. Finally, they empirically investigated the data reduction for bug triage in bug repositories of two large open source projects, namely Eclipse and Mozilla.In our paper, we propose an alternative framework called Auto-BugTriager that aims to alleviate the problem of data reduction. Which consist of three phases namely InfoZie, DataReduction and NBPredictor which process together to predict the recommendation list of expert developer for fixing the bug.

Nicolas Bettenburg et al. [2] developed a tool called InfoZilla which compares the structural elements extracted from the master and duplicate bug reports. The description of bug reports are typically treated as natural language text, although it often contains stack traces, source code, and patches.

Ignoring such structural elements is a loss of valuable information; these structural useful information's helps in improving the performance of machine learning approaches as well as the accuracy of bug triage. Also, as specified in [3] the duplicate reports in the bug repositories are not always a copy of original. Most of the times these duplicate reports are re-submitted intentionally by the programmers. There are 90% of chances for the presence of extra valuable information's in the duplicate reports. Removing such valuable information will decrease the accuracy of bug triage.

Therefore, it is important to validate the duplicate reports available in the bug repository. The validation can be performed using a specific tool that as in [2] called InfoZilla which validates the duplicate reports through the extracted structural information from the bug reports.In proposed system, we create a tool called InfoZie based on diff algorithm (diff is a data comparison tool used to show the changes between two versions of the same file) that compares the technical elements extracted from both duplicate and master bug reports. When duplicate report contains extra information's, it is merged with the master bug report otherwise ignored. This improves the bug dimension and also the accuracy of bug triage buy handling the duplicate bug reports.

V. Bolon-Canedo et al. [4], projected the experimental analysis of several fundamental algorithms which has been studied to assess their performance in a controlled scenario. Out of all, Feature Selection Algorithm is increasingly reliable with sample size and pursue the solution of a clearly stated optimization goal. Therefore, in proposed system we use feature extraction method to remove the unwanted stop words from the bug reports. This helps improve the quality of dataset as well as improve the accuracy of bug triage.

D. Cubranic et al. [5] developed an application of supervised machine learning (text categorization) has been developed and using a naive Bayes classifier to automatically assign bug reports to developers. Only 30 percent of classification accuracy is achieved with the demonstration of 15,859 bug reports from a large open-source project, Eclipse.org, The analysis shows that, Naive Bayes classifier algorithm cannot deal with unlabelled documents in the corpus.

In our paper we propose to combine Expectation Maximization (EM) method with Naïve Bayes (NB) classifier to achieve very good results in classifying a document corpus that containing a high proportion of unlabelled documents.

T. M. Khoshgoftaar et al. [6] presents a process involving a feature selection technique for selecting the important attributes and a data sampling technique for addressing class imbalance. The study answers research questions such as whether to apply feature selection to the original or sampled data, and whether the training data should be formed based on the original or sampled data, given a set of selected features. In our paper, as specified in [1] we extract attributes from the history of existing bug reports based on the technical aspect as well as developers detail to train the classifier.

J. Anvik et al. [7] projected a semi-automated approach which can be used to ease one part of bug triage process, i.e., the assignment of reports to a developer by using a supervised machine learning algorithm. The system has been demonstrated for the Eclipse and Firefox projects, which achieved precision rates of over 50%, reaching 64% respectively. But when the system is applied for gcc project, the results were far worse i.e., only 6% of accuracy is achieved because of characteristics of the project. And, S. Kim et al. [8]presents a bug finding algorithm using bug fix *memories*: a project-specific bug and fix knowledge base developed by analyzing the history of bug fixes. Analysis of five open source projects shows that, for these projects, 19.3%-40.3% of bugs appear repeatedly in the memories. The results analysis demonstrates that project-specific bug fix patterns occur frequently enough to be useful as a bug detection technique.In our paper we aim to automate the bug triage in a domain specific environment. As every software companies spends 40% of their time and cost on manual bug triage.

Bug tracking systems play a central role in supporting collaboration between the developers and the users of the software. As in [9] highlights the importance of user community involvement in bug fixing activities, and keeping them up-to-date about the status of a bug.We propose to create a user friendly application in java, i.e., in a platform independent environment and also we tend to add extra supporting features to the application system which will help both the developer and user to manage and organize the bug reports.

3. Proposed System

The proposed system design a framework called Auto-BugTriager as shown in Figure 1. Which aims to eliminate the problem of data reduction in bug triage. This framework consist of three phases, namely InfoZie, DataReduction and NBPredictor which process together to predict the recommendation list of expert developer for fixing the bug. The processing of each phase is as described below.

3.1. InfoZie

This phase creates a comparative tool that validates the duplicate reports by comparing the extracted technical terms from both master and duplicate bug reports.

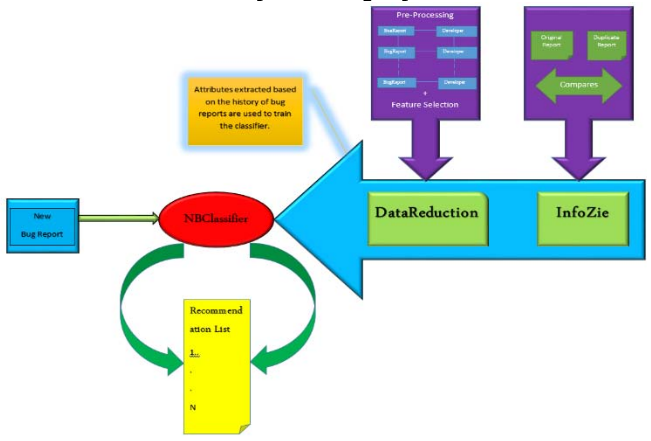

Figure 1: Auto-BugTriage Framework

3.2. DataReduction

This is the second phase that performs the following two process.

3.2.1. Pre-processing

Here, text classification technique is used to convert the summarized bug reports into text matrix where each row indicates one bug report, and each column indicates one word.

3.2.2. Noise Reduction

This phase make use of feature extraction method to reduce the word dimension i.e., removes the uninformative stop words from the bug reports.

3.3. NBClassifier

This is the final phase of the framework that builds a Naïve Bayes classifier trained with the attributes extracted from the history of processed bug reports. The NBclassifier phase predicts the expert recommendation list for fixing the new bug.

We propose to extend this idea for real-time use by implementing the Auto-BugTriager framework at backend process of an application system. We implement the system in java using Eclipse platform to as it provide a platform independent environment to run the application system. This system helps the software developing team in managing and organizing the bug reports by adding extra supporting features.

4. Conclusions

One of the time-consuming step in processing bug triage is to assign an appropriate developer to fix a new bug. Data reduction is one of the major problem identified in applying text classification to automate the bug triage process, which decreases the accuracy of bug triaging. This paper explains the complete idea behind the automatic bug triage process, and proposed a framework called Auto-BugTriager that eliminates the problem of data reduction to greater extent in bug triaging. Our framework consist of three phases namely InfoZie, DataReduction and NBPredictor which process together to predict the recommendation list of expert developer for fixing the bug. The proposed system can be used for any all kinds of domain specific project environment that generate large number of bug data every day. This proposed system will help software companies in improving both the software quality and accuracy of bug triage.

References

[1] JifengXuan, He Jiang, Yan Hu, ZhileiRen, WeiqinZou, ZhongxuanLuo, 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] Nicolas Bettenburg and Rahul Premraj, "Extracting Structural Information from Bug Reports," May 2008,

[3] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful ... really?" in ICSM08: Proceedings of IEEE International Conference on Software Maintenance, 2008, pp. 337–345.

[4] V. Bolon-Canedo, N. Sanchez-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.

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

[6] T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction," in Proc. 22nd IEEE Int. Conf. Tools Artif. Intell., Oct. 2010, pp. 137–144.

[7] 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.

[8] S. Kim, K. Pan, E. J. Whitehead, Jr., "Memories of bug fixes," in Proc. ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2006, pp. 35–45.

[9] 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.