International Journal of Research in Advent Technology (IJRAT)
(E-ISSN: 2321-9637)
Special Issue National Conference "CONVERGENCE 2017", 09th April 2017

Towards Effective Bug Triage with Software Data Reduction Techniques

[1] Ms.P.S.Kharche, [2]Dr.P.M. Jawandhiya
[1]P.G. Student of CSE, PLITMS, Buldana, India
[1]kharche.priyanka8@gmail.com
[2]Principal,PLITMS, India

Abstract -Bug triage is an important step in the process of bug fixing. The goal of bug triage is to assign a new-coming bug to the correct potential developer. The existing bug triage approaches are based on machine learning algorithms, which build classifiers from the training sets of bug reports. As per literature, need to develop a effective model for doing data reduction on bug data set which will reduce the scale of the data as well as increase the quality of the data., by reducing the time and cost. Many software companies spend their most of cost in dealing with these bugs. We are using instance selection and feature selection simultaneously with historical bug data. To decrease the manual and time cost, text classification techniques are applied to perform the automatic bug triage.

1. INTRODUCTION

A bug repository plays an important role in managing software bugs. Many open source software projects have an open bug repository that allows both developers and users to submit defects or issues in the software, suggest possible enhancements, and comment on existing bug reports. The problem is caused by inadequate or invalid logic. A bug can be an error, mistake, flaw or fault, which may cause collapse or variation from usual results. Most bugs are due to human errors in source code or its design. Software companies spend over 45 percent of cost in fixing bugs .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. In modern software development, software repositories are large-scale databases for storing the output of software development, e.g., source code, bugs, emails, and specifications. By leveraging data mining techniques, mining software repositories can uncover interesting information in software repositories and solve real world software problems. A bug repository (a typical software repository, for storing details of bugs), plays an important role in managing software bugs.. In this paper, bug reports in a bug repository are called bug data. 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. It is a challenge to manually examine such large-scale bug data in software development. 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. In traditional software development, new bugs are manually triaged 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 in time cost and low in accuracy. To avoid the expensive cost of manual bug triage, existing work [1] has proposed an automatic bug triage approach, which applies text classification techniques to predict developers for bug reports. Based on the results of text classification, a human triage assigns new bugs by incorporating his/her expertise. However, large-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage. In this paper, we address 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.

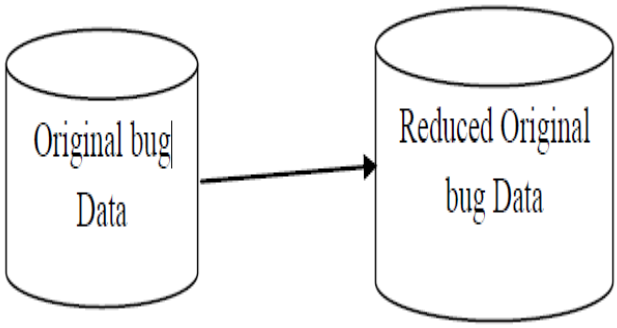

Figure 1: Bug Data Reduction

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. In our work, we combine existing techniques of instance selection and feature selection to simultaneously reduce the bug dimension and the word dimension. The reduced bug data contain fewer

bug reports and fewer words than the original bug data and provide similar information over the original bug data. We evaluate the reduced bug data according to two criteria: the scale of a data set and the accuracy of bug triage. To avoid the bias of a single algorithm, we empirically examine the results of four instance selection algorithms and four feature selection algorithms. Given an instance selection algorithm and a feature selection algorithm, the order of applying these two algorithms may affect the results of bug triage. In this paper, we propose a predictive model to determine the order of applying instance selection and feature selection. We refer to such determination as prediction for reduction orders. Drawn on the experiences in software metrics,1 we extract the attributes from historical bug data sets. Then, we train a binary classifier on bug data sets with extracted attributes and predict the order of applying instance selection and feature selection for a new bug data set. Experimental results show that applying the instance selection technique to the data set can reduce bug reports but the accuracy of bug triage may be decreased; applying the feature selection technique can reduce words in the bug data and the accuracy can be increased. A Software bug is an issue causing a program to collapse or create unacceptable output. Due to huge number of daily bugs and lack of skill person of all the bugs, manual triage is an expensive in time cost and labor cost, low in precision.

2. LITERATURE REVIEW

Sr. No	**Title of Paper**	**Publication Details**	**Conclusion / Highlights of Related Paper Point wise**	**Limitation Point wise in respective paper**	**Remark / Drawbacks / Future Scope**
1.	Finding bugs in Web applications using dynamic test generation and explicit-State model checking	Author: Adam ki ezun, Julian Dolby, Frank Tip, Danny Dig, Year**:** 2010	find two kinds of failures in web applications: 1. Execution failures that are manifested as crashes or warnings during program execution. 2. HTML failures that occur when the application generates malformed HTML	It cannot handle the dynamically generated pages that are ubiquitous	We perform an automated analysis to minimize the size of failure-inducing inputs.

2	Assisting Bug Rep ort Triage through Recommendation	Author: John Karsten Anvik Year: 2007	1. Investigated four types of RDO recommenders: a recommender for who to assign a rep ort to (RA), recommenders for which component (RC) and sub-component (RS) to file a report against, and a recommender for which other project members may want to be informed about progress on this report (RI). 2. Provides a study of how triage is accomplished and the challenges faced by triagers for the project team of a successful open-source project.	The appropriate number needs to be known in advance. Different projects have different quantities of reports	We further evaluated the RA, RC, RS, and RI recommenders in a field study conducted with four Eclipse platform triagers for the UI component
3.	Towards graphical models for text processing	Author: Charu C. Aggarwal Peixiang Zhao Year 2013	I. The goals is to design a representation which retains at least some of the ordering information among the words in the document without losing its flexibility and efficiency for data processing. I. The distance graph representation maintains information about the relative placement of words with respect to each other.	It will not explore the undirected variation too extensively	This approach enables knowledge discovery from text which is not possible with the use of a pure vector-space representation, because it loses much less information about the ordering of the underlying words.

4.	Hyper-Quad-Tree based K-Means Clustering Algorithm for Fault Prediction	Author: Swati Varade, Madhav Ingle Year: 2013	This paper illustrates hyper quad tree based k-means algorithm for software fault prediction. 1. This system overcomes the weaknesses in k-means algorithm using Hyper Quad Tree . 2. Hyper quad tree works in n-dimensions hence it finds better initial cluster centers than former algorithms.	The user has to initialize the number of clusters which are very not easy to identify in the majority of the cases.	focused on automatic initialization of number of clusters now which is manually initialized by the user

3. RELATED WORKS

As our Knowledge, there is no combination of data reduction methods in turn to decrease the data scale and upgrade the exactness of bug triage approach in the illustration. Fu.Y, Zhu.X, and Li.B [4] investigated to obtain the accurate prediction model with minimum cost by labeling most informative instances. In contrast to these papers, our paper aims to employ the information gain algorithm to develop the software value of bug data prediction. In this paper, we focus on the issue of bug data reduction and low in precision of bug data set. Once a software bug is found, a reporter (typically a developer, a tester, or an end user) records this bug to the bug repository. In a bug report, the summary and the description are two key items about the information of the bug, which are recorded in natural languages. As their names suggest, the summary denotes a general statement for identifying a bug while the description gives the details for reproducing the bug. A developer, who is assigned to a new bug report, starts to fix the bug based on the knowledge of historical bug fixing [6], [6]. Presented approach is based on a supervised machine learning algorithm that is applied to information available in the bug repository. When a new report arrives, the classifier produced by the supervised machine learning technique offered a small number of developers suitable to resolve the report. Typically, the developer pays efforts to understand the new bug report and to examine historically fixed bugs as a reference (e.g., searching for similar bugs [5] and applying existing solutions to the new bug [8]). An item status of a bug report is changed according to the current result of handling this bug until the bug is completely fixed. Changes of a bug report are stored in an item history. This bug has been assigned to three developers and only the last developer can handle this bug correctly. Changing developers lasts for over seven months while fixing this bug only costs three days.

4. DATA REDUCTION FOR BUG TRIAGE USING

A.INSTANCE 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. In our work, we leverage the combination of instance selection and feature selection to generate a reduced bug data set. We replace the original data set with the reduced data set for bug triage. 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) [18] while feature selection aims to obtain a subset of relevant features (i.e., words in bug data) [19]. In our work, we employ the combination of instance selection and feature selection. To distinguish the orders of applying instance selection and feature selection, we give the following denotation. Given an instance selection algorithm IS and a feature selection algorithm FS, we use FS! IS to denote the bug data reduction, which first applies FS and then IS; on the other hand, IS! FS denotes first applying IS and then FS.

- Instance selection methods associated with data mining tasks such as classification and clustering

- It's a nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. Choosing a subset of data to achieve the original purpose of a data mining application as if the whole data is used.
- The ideal outcome of instance selection is model independent.

B.FEATURE SELECTION

- It select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features [1].
- Reduce # of patterns in the patterns, easier to understand.
- Create new attributes that can capture the important information in a data set much more efficiently than the original attributes.
- Use the smallest representation which is enough to solve the task.

5. CONCLUSION AND FUTURE WORK

This paper is the first work of combining feature selection with instance selection to reduce the training set for the bug triage problem. The motivation of this work is to reduce the large scale of the training set and to remove the noisy and redundant bug reports for bug triage.The experimental results show that the combinations of CHI and ICF can achieve better accuracy rates than that without the training set reduction. The results also indicate that the combination, ICF→CHI, is a good choice for the training set reduction. In the future work, we plan to propose a unified approach to merge the tasks of feature selection and instance selection. In this paper, we focus on the combinations of the existing algorithms for the training set reduction. Since each algorithm in the combination is limited by the other one, it is necessary to develop a unified approach to integrate feature selection and instance selection. Another future work is to apply the training set reduction of bug triage to other tasks to improve the software quality. Since machine learning becomes one of the powerful tools in software engineering, the training set reduction can be useful for the work based on machine learning.

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] J. Anvik, "Automating bug report assignment," Proc. Intl. Conf.Software Engineering (ICSE 06), ACM, May 2006, pp. 937-940.

[3] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?,"Proc. Intl. Conf. Software Engineering (ICSE 06), ACM, May 2006,pp. 361-370.

[4] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

[5] 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.

[6] 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.

[7] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng., May 2010, pp. 481–490.

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

[9] 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 explicitstate model checking," IEEE Softw., vol. 36, no. 4, pp.474–494, Jul./Aug. 2010.

[10] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2011.