METHODOLOGY TO AUTOMATE THE BUG REPORT CLASSIFICATION: A STUDY

Ashish Kumar Sharma*

Abstract

Bug reports are most important for the maintenance of software. In this era, open-source software is enormous and used throughout the world. The users of software are developers, testers, and end-users. So the bug reports were submitted by a variety of people from worldwide, for the same problem the many users sent the bug reports, and the bug repository having duplicate reports for the same problem. To recognize the duplicate bug reports researcher performed different approaches to process the reports. A review study of the research work done on bug reports classification to identify the duplicate bug reports. In this paper firstly present the background detail for bug reports and reports on bug repository to show the importance of bug reports classification. Then to discuss the issues in bug reports classification and a summary of existing work on bug report classification.

Keywords: *Bug Report, Bug-report Analysis, Bug-report Triage, Bug Localization, Bug Fixing, Bug Life Cycle.*

Introduction

Software developers giving their full efforts but software having bugs that are identified at run time. Software maintenance is more costly in comparison to other phases of the software development life cycle. The testing and maintenance are estimated that cost one-third of the total cost of software [1, 2]. To ensure software quality and efficiency many software projects use a bug reports repository to collect and store bug reports submitted by developers, testers, and end-users.

**Research Scholar, Department of Computer Science & Engineering, Indian Institute of Technology, Banaras Hindu University, Varanasi (Uttar Pradesh)*

Since the software is used worldwide so a vast amount of bug reports is submitted in the bug repository. These bug reports are very helpful in the quality improvement of software projects but further, due to their large size, it becomes difficult to manage and solve these reports one by one.

Background

Software products are growing in number and size both, which make the maintenance process very challenging. Maintenance of software is the most time taking process during the life cycle of a software system.

The most important actors involved in maintenance are bug reporting and triaging. For large software systems used worldwide during maintenance, bug reporting required a huge database. Since the bug reports data set are large in amount so it required a lot of time to find the bug report to assign the trigger predict them and fix them when they appear. For any software system, it's most important to be conscious of bug reports that when and how the bugs are to be fixed. The developers are specific to technology so assigning bugs to an appropriate developer will take time if the trigger failed to assign the bugs to the proper developer then the developer wastes a lot of time in solving these bugs.

The bug reports are in text data, in natural languages so recognizing the format of the report and content of the report is very tedious. The user of a software system may be a developer, a tester, or a user so for the same problem each can define the problem in different ways, and most importantly for the same problem different people submit individual bug reports so the system becomes bulkier and after solving the bug report the developer find that these problems are solved and during this, the bug reports are which severe in nature that must solve on priority over other bug reports. So for efficient maintenance of software products, the bug report must be analyzed in a short time that to prioritize the bug reports on a severity basis and identifying the bug reports that addressing the same problem so that the time of trigger and developer both we can save. A bug report is a textual report that may be structured containing several fields i.e. ID, TITLE, SUMMARY, AND DESCRIPTION, etc.

Table 1: Many Duplicate Reports Submitted for Chromium Project

SN	ID	Summary
1	131686	Chrome failed to display text in some PDF files
	130952	**PDF Text Invisible, able to be moused over and selected, but, still not visible**
2	39076	REGRESSION: Full-screen mode does not size Aero glass frame correctly
	37588	**REGRESSION: Wrong client area for web pages in glass frame full-screen mode**
	38805	**Window borders are still visible in a full-screen application shortcut window**
3	44789	Bookmark manager comes up blank
	43448	**Bookmark Manager doesn't work if the process limit is exceeded (shows blank / empty tab**
	44807	**Bookmark Manager doesn't show any bookmark**
	46006	**Bookmark manager may start showing blank on log out and re-login**

History

There are approaches proposed by a researcher to manage the large dataset of bug reports. A bug rule-based classification technique to categorize bug reports. After categorizing the bug report's authors provided a feedback mechanism, which distinguished duplicate and valid bug reports. [1] Predicting the severity of a reported bug by analyzing its textual description using text mining algorithms. The author considered the three cases i.e. Mozilla, Eclipse, and GNOME [2]. Bug reporting, bug reporters may often mislabel SBRs as NSBRs partly due to lack of security

domain knowledge so the author developed a new approach that applies text mining on natural language descriptions of BRs to train a statistical model on already manually-labelled BRs to identify SBRs that are manually-mislabelled as NSBRs. [3].To merge bug duplicates, rather than treating them separately. They quantify the amount of information that is added for developers and show that automatic triaging can be improved as well [4]. Leverage recent advances on using discriminative models for information retrieval to detect duplicate bug reports more accurately. They have validated their approach on three large software bug repositories from Firefox, Eclipse, and Open Once. [5]

Table 2: Tabular Integrated Summary of Literature Studied

Paper-No	Data-Set	Data-Source	Tools	Technique	Result (precision/recall)
1	-	Bugzilla-Server	Self-Crawler	Rule-based classification	Precision of 0.9
2	400	Mozilla	Bugzilla	Naïve-Bayes Classification	0.65-0.75 with Mozilla and Eclipse; 0.70-0.85 with GNOME
	200	Eclipse			
	450	GNOME			
3	10000000 SLOC	CISCO Project	SAS	text-mining	78
4	211843	Eclipse	infoZilla	SVM/Naïve Bayes	65
5	12732	Open Office	-	SVM	17-31
	44652	Eclipse			35-43
	47704	Firefox			22-26

State of the Art

The topic modelling on the corpora of processed bug reports of open-source software projects HTTP Client, Jackrabbit, and Lucene the machine learning approaches decision tree, logistic regression and naive Bayes classifier **[6].** Extracting the old bug reports from the bug report database and identifying the problem type and giving them a topic name. On arrival of new bug reports the model match with the existing old bug and then the severity of the bug is defined **[7].** Firstly pre-process the data for feature selection and then applying text mining approaches on bug databases accumulo, ActiveMQ, camel, flume, and wicket **[8].** The effectiveness of various supervised learning algorithms to predict if a bug report reopened, classical supervised learning algorithm in machine learning literature, i.e., kNN, SVM, Simple Logistic, Bayesian Network, Decision Table, CART, and LWL, and 3 ensemble learning algorithms, i.e., AdaBoost, Bagging and Random Forest TRAM (TRiaging Approach using bug reports Metadata) perform the experimental evaluation on open-source projects namely Free desktop, Net Beans, Eclipse, and Firefox. **[9].**Different probabilistic techniques classify reviews into four types: bug reports, feature requests, user experiences, and ratings. The review metadata such as star rating and the tense, as well as, text classification, natural language processing, and sentiment analysis techniques. When the metadata is combined with the natural language processing the classification result is better**[10].** A novel approach uses a deep neural network (DNN) in combination with rVSM, an information retrieval (IR) technique. rVSM collects the feature on the textual similarity between bug reports and source files. DNN is used to learn to relate the terms in bug reports to potentially different code tokens and terms in source files and documentation if they appear frequently enough in the pairs of reports and buggy files. **[11].**

Table 3 : Tabular Integrated Summary of Recent Methodology and Approaches

Paper-No.	Data-Set	Data-Source	Tools	Technique	Result (precision/recall)
6	30,000	Eclipse, Mozilla, and Netbeans	Latent Dirichlet Allocation (LDA)	Stanford Topic Modelling Toolbox (TMT)	F-a measure of 70%.
7	3,203	accumulo, activemq, camel, flume, and wicket	naive Bayes, SVM, naive Bayes multinomial	-	F-measure are 0.811, 0.450, and 0.880

8	25154	Freedesktop, NetBeans, Eclipse, and Firefox	TRAM (TRiaging Approach using bug reports Metadata)	-	F-score by approximately 34%, 40%, 20%, and 21%

Conclusion

Bug report analysis consists of a lot of research work. We discussed major statistics on bug reports how important duplicate bug report identification is for bug triaging as well as bug localization. Then we study several research work done on duplicate bug reports that mainly use Data Mining Machine Learning and Information Retrieval methods for bug-report analysis, bug-report optimization, bug-report triage, and bug fixing. However, several tasks of duplicate bug reports are not solved and we are trying to automate the whole process. All researchers are focusing on automating duplicate bug reports (e.g., bug-report assignment, duplication detection, bug localization). That is, it is hard to apply existing automatic approaches in practice. Furthermore, bug-tracking systems are still not able to provide any support for automatically dealing with bug reports. In the future, researchers may consider how to improve the accuracy of existing automatic approaches.

References

1. *Battenburg, N., Premraj,R. Zimmermann, T. and Kim, S. (2008), "Duplicate Bug Reports Considered Harmful... Really?" Proc. IEEE Conf. Software Maintenance (ICSM 08), IEEE, pp. 337-345, September.*
2. *Lamkanfi, Demeyer,S., Giger, E. and Goethals, B. (2010), "Predicting the Severity of a Reported Bug" in 2010 7th IEEE Working Conference on Mining Software Repositories (MSR), IEEE, pp.1–10.*
3. *Lam, N.A., Nguyen, T., Nguyen, H. A. and Nguyen, T. N. (2015), "Combining deep learning with Information retrieval to localize Buggy files for Bug reports (n)," in ASE'15: Proc. of the International Conference on Automated Software Engineering*.
4. *Maalej, W. and Nabil, H. (2015), "Bug Report, Feature request, or simply praise? On Automatically Classifying App Reviews", in 2015 IEEE 23rd International Requirements Engineering Conference (RE), pp. 116–125, IEEE.*

5. *Michael, G. Rotella, P. and Xie, Tao (2010), "Identifying Security Bug Reports via Text Mining: An Industrial Case Study,"Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on IEEE.*

6. *Pingclasai, Hata, H. and Matsumoto, K. (2013), "Classifying Bug Reports to Bugs and other requests using topic Modelling," in Proceedings of 20th Asia-Pacific Software Engineering Conference (APSEC), vol. 2, pp. 13–18, IEEE.*

7. *Sun, C., Lo, D., Wang, Wang, X., Jiang, J., and Khoo, Siau-Cheng (2010), "A Discriminative Model Approach for Accurate Duplicate Bug Report Retrieval", Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE 10), pp.45-54.*

8. *Xia, X., Lo, D., Qiu, W., Wang, X. and Zhou, B. (2014), "Automated Configuration Bug Report Prediction using Text Mining, in "COMPSAC. IEEE, pp. 107–116.*

9. *Xia, X., Lo, D., Wang, X., Yang, X., Li, S. and Sun, J. (2013), "A comparative study of supervised Learning Algorithms for reopened Bug prediction," in Software Maintenance and Reengineering (CSMR), 17th European Conference on IEEE, pp. 331–334.*

10. *Yang, G., Zhang, T. and Lee, B. (2014), "Towards Semi-automatic Bug Triage and Severity Prediction based on topic Model and Multi-feature of Bug Reports. In Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference, COMPSAC'14, pages 97–106, Washington, DC, USA, IEEE Computer Society.*

11. *Zhang, T. and Lee, Byung Jeong (2011), "A Bug Rule Based Technique with Feedback for Classifying Bug"Reports, Proceedings of the 2011 IEEE 11th International Conference on Computer and Information Technology, pp.336-343, August 31-September 02.*

***