ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Survey on: An Effective and Accurate Bug Triage System with Software Data Reduction Techniques

Varsha Bongane [1], S.B.Natikar [2]

P.G. Student, Department of Computer Engineering, VACOE, Ahmednagar, Maharashtra, India[1]

Associate Professor, Department of Computer Engineering, VACOE, Ahmednagar, Maharashtra, India[2]

ABSTRACT: The process of fixing bug is bug triage or bug assortment. The aim of this, to correctly assign a developer to a new bug. Triaging these incoming reports manually is error-prone and time consuming.Software companies pay most of their cost in dealing with these bugs. For software repositories traditional software analysis is not completely suitable for the large-scale and complex data. To reduce time and cost of bug triaging,present an automatic approach to predict a developer with relevant experience to solve the new coming report. In proposed approach explain data reduction on bug data set which will reduce the scale of the data as well as increase the quality of the data. And also give domain specific bugs with their solution by developers. For implementing this use instance selection and feature selection for reducing bug of data. And Top-K pruning algorithms for tackling domain specific task.

KEYWORDS: Bug, Bug Triage, repositories, instance selection.

I. INTRODUCTION

Numerous product organizations spend a large portion of the cash in settling the bugs. Substantial programming ventures have bug archive that gathers all the data identified with bugs. In bug store, every product bug has a bug report. The bug report comprises of printed data with respect to the bug also, redesigns identified with status of bug settling. Once a bug report is framed, a human triager doles out this bug to a designer, who will attempt to settle this bug. This designer is recorded in a thing relegated to. The assignedto will change to another designer if the beforehand alloted engineer can't alter this bug. The procedure of doling out a right designer for settling the bug is called bug triage. Bug triage is a standout amongst the most tedious venture in treatment of bugs in programming ventures[1].

Manual bug triage by a human triager is tedious what's more, blunder inclined following the quantity of day by day bugs is substantial and absence of learning in designers about all bugs. Due to every one of these things, bug triage results in costly time misfortune, high cost and low precision[2]. The data put away in bug reports has two principle challenges. Firstly the substantial scale information and furthermore low nature of information. Because of huge number of day by day reported bugs, the quantity of bug reports is scaling up in the vault. Uproarious and excess bugs are debasing the nature of bug reports. In this paper a powerful bug triage framework is proposed which will lessen the bug information to spare the work expense of designers. It likewise means to fabricate a fantastic arrangement of bug information by expelling the repetitive and non-enlightening bug reports[3,4].

I.A) Objectives:

1) Simultaneously reduce the scales of the bug dimension and the word dimension.
2) Improve the accuracy of bug triage.
3) Improve the results of data reduction in bug triaging to explore how to prepare a high quality set of bug data and tackle a domain specific task.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

II. RELATED WORK

In [1] they mention that Bug triaging is an error-prone, tedious and time consuming task. They are going with Revisiting Bug Triage and Resolution Practices. In this paper they studied about bug triaging and fixing practices, including bug reassignments and reopenings, in the context of the Mozilla Core and Firefox projects, which they consider to be representative examples of a large-scale open source software project. Also they have plan to conduct qualitative and quantitative analysis of the bug assignment practices. We are interested in providing insights into several areas: triage practices, review and approval processes; root cause analysis of bug reassignments and reopens in open source software projects; and recommendations for improvements/redesign of bug tracking systems.
In [2] this paper, they introduce a graph model based on Markov chains, which captures bug tossing history. This model has several desirable qualities. First, it reveals developer networks which can be used to discover team structures and to find suitable experts for a new task. Second, it helps to better assign 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 [3] recent research shows that optimizing recommendation accuracy problem and proposes a solution that is essentially an instance of content-based recommendation (CBR). However, CBR is well-known to cause over-specialization, recommending only the types of bugs that each developer has solved before. This problem is critical in practice, as some experienced developers could be overloaded, and this would slow the bug fixing process. In this paper, they take two directions to address this problem: First, we reformulate the problem as an optimization problem of both accuracy and cost. Second,we adopt a content-boosted collaborative filtering (CBCF), combining an existing CBR with a collaborative filtering recommender (CF), which enhances the recommendation quality of either approach alone.
In [4] Current techniques either use information retrieval and machine learning to find the most similar bugs already fixed and recommend expert developers, or they analyze change information stemming from source code to propose expert bug solvers. Neither technique combines textual similarity with change set analysis and thereby exploits the potential of the interlinking between bug reports and change sets. In this paper, they present our approach to identify potential experts by identifying similar bug reports and analyzing the associated change sets. Studies have shown that effective bug triaging is done collaboratively in a meeting, as it requires the coordination of multiple individuals, the understanding of the project context and the understanding of the specific work practices. Therefore, they implemented approach on a multi-touch table to allow multiple stakeholders to interact simultaneously in the bug triaging and to foster their collaboration.

II.A) PROBLEM STATEMENT

The problem statement is as follows:

- Traditional software analysis is not completely suitable for the large-scale and complex data in software repositories.
- In traditional software development, new bugs are manually triaged by an expert developer i;e, a human triager. Due to 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.
- The number of regular occurring bugs for open source large-scale software project is so much large that makes the triaging process very difficult and challenging.
- To decrease the time cost in manual work, text classification techniques are applied to conduct automatic bug triage.

III. PROPOSED SYSTEM

Triage is a system used by software development teams to ration limited technical resources when the number of defects needing resolution exceeds the resources available to correct and verify them so as to resolve the greatest number of defects possible. If there is a concept that testers and test managers are acutely familiar with, it's "limited resources." Unfortunately we can't fix and retest everything in the limited amount of time we have or with current resources[6,7]. Bug triage is a formal process where each bug is prioritized based on its severity, frequency, risk and

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

etc. to achieve better balance working on the important and unimportant bugs. This process makes time save as we need not worry much about the less important bugs first[1].

III.A) SYSTEM ARCHITECTURE

a) Bug Triage

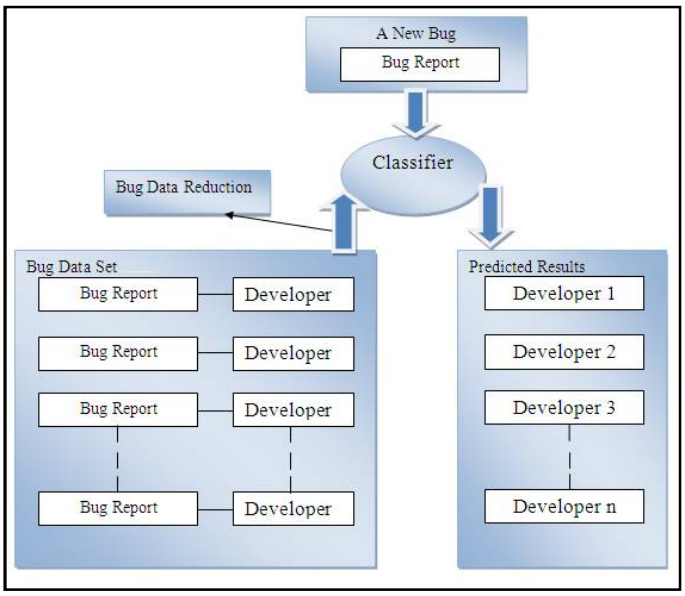
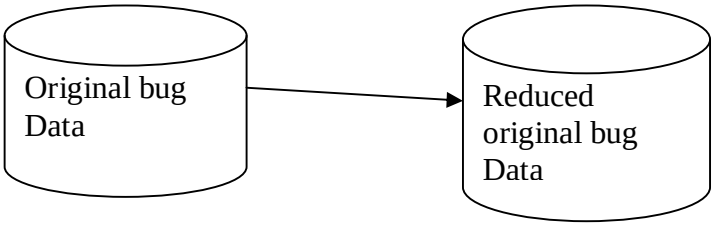


Fig.1 System Architecture

The diagram in figure 1 illustrated the system architecture of the proposed system. The input to the system is in the form of bug data set. The bug data set consists all the details of software bugs. Each bug has bug report and the details of the developer who have worked on that respective bug[6]. The bug report is mainly divided in two parts, summary and description. The proposed system gives predicted results in form of output. Basically, there are two types of users in the proposed system. First is the developer and second is the tester. Developer will get software bugs assigned to him. Developer can work on only one software bug at a time. Tester can add new bugs to the system.

b) Data Reduction

Here we are reducing the bug data by using instance and feature selection so that we get low scale as well as quality data.



ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

IV. ALGORITHM

Instance and Feature Selection. FS → IS = Bug data reduction. which first applies FS and then IS. On the other hand, IS → FS denotes first applying IS and then FS In Algorithm 1, we briefly present how to reduce the bug data based on FS → IS. Given a bug data set, the output of bug data reduction is a new and reduced data set.

Algorithm 1. Data reduction based on FS →IS

Input: training set *T* with *n* words and *m* bug reports,
reduction order FS→IS
final number *n F* of words,
final number *m I* of bug reports,
Output: reduced data set *T FI* for bug triage
1) apply FS to n words of T and calculate objective values for all the words;
2) select the top *n F* words of *T* and generate a training set *T F*;
3) apply IS to *m I* bug reports of *T F*;
4) terminate IS when the number of bug reports is equal to or less than *m I* and generate the final training set *T FI.*

For the combinations of *FS* and *IS* , we will present the results of *FS→IS* and *IS→FS* , respectively. All the combination results are shown in details in the experiment section. In the following parts of this section, we briefly introduce the feature selection algorithm CHI and the instance selection algorithm ICF used in our work[1,6].

Two algorithms FS and IS are applied sequentially Note that in Step2, some of bug reports may be blank during feature Selection In our work, FS → IS and IS → FS are viewed as two orders of bug data reduction. To avoid the bias from a single algorithm, we examine results of four typical algorithms of instance selection and feature selection, respectively.
1) Iterative Case Filter (ICF)
2) Learning Vectors Quantization (LVQ)
3) Decremental Reduction Optimization Procedure (DROP)
4) Patterns by Ordered Projections (POP)

V. CONCLUSION AND FUTURE WORK

This paper presented an all-inclusive survey on the data reduction technique for bug triage. The main features, the advantages and disadvantages of each technique are described. As Bug triage is a vital step of software maintenance in both labor cost and time cost. The goal is to correctly assign a developer to a new bug for further handling. Many software companies spend their most of cost in dealing with these bugs. 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. As per survey, there is strong need to focus on reducing bug data set in order to have less scale of data and quality data. Propose the improved feature selection method by using kruskal model for addressing the problem of data reduction.
In future work, we plan to propose a unified approach to merge the tasks of feature selection and instance selection. Our future plan also includes an empirical study of the use of the approach by bug triagers on an open source system, an investigation of additional sources of information.

REFERENCES

1. Olga Baysal, Reid Holmes, and Michael W. Godfrey David R. Cheriton ,"Revisiting Bug Triage and Resolution Practices" School of Computer Science University of Waterloo Waterloo, ON, Canada {obaysal, rtholmes, migod}@uwaterloo.ca.
2. Graphs Gaeul Jeong ,"Improving Bug Triage with Bug Tossing ", Seoul National University gejeong@ropas.snu.ac.kr.
3. Jin-woo Park, Mu-Woong Lee, Jinhan Kim, Seung-won Hwang, "COSTRIAGE: A Cost-Aware Triage Algorithm for Bug Reporting Systems" POSTECH, Korea, Republic of,jwpark85,sigliel,wlsgks08,swhwang}@postech.edu
4. Katja Kevic, Sebastian C. Muller, Thomas Fritz, and Harald C. Gall,"Collaborative Bug Triaging using Textual Similarities and Change Set Analysis," Department of Informatics University of Zurich, Switzerland katja.kevic@uzh.ch {smueller, fritz, gall}@ifi.uzh.ch

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

5. Jifeng Xuan1 He Jiang2, 3 Zhilei Ren1 Jun Yan4 Zhongxuan Luo1,"Automatic Bug Triage using Semi-Supervised Text Classification", 21 School of Mathematical Sciences, Dalian University of Technology, Dalian, 116024 China 2 School of Software, Dalian University of Technology, Dalian, 116621 China 3 State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, 100190 China 4 Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing, 100190 China 1.
6 Ji feng Xuan, He Jiang, Member, IEEE, Yan Hu, Zhilei Ren, Weiqin Zou, "Towards Effective Bug Triage with Software Data Reduction Techniques" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 1, JANUARY 2015.
7. J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in Proc. 28th Int. Conf. Softw. Eng.,, pp. 361–370, May 2006
8. 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.
9. 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 textprocessing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.
11. 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.
12. V. Bol_on-Canedo, N. S_anchez-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.
13. A. K. Farahat, A. Ghodsi, M. S. Kamel, "Efficient greedy feature selection for unsupervised learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 285–310, May 2013.
14. 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.
15. Q. Hong, S. Kim, S. C. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in Proc. 27th IEEE Int. Conf. Softw. Maintenance, pp. 323–332, Sep. 2011.
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.
17. Q. Hong, S. Kim, S. C. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in Proc. 27th IEEE Int. Conf. Softw. Maintenance, pp. 323–332, Sep. 2011.

BIOGRAPHY

Miss.Varsha T.Bongane received the B.E.(Information Technology) from Savitribai Phule Pune University in 2014 and pursuing M.E. degree in Computer Engineering from Vishwabharati Academy's College of Engineering, Savitribai Phule Pune University,Ahmednagar, Maharashtra, India in 2015-16.

Prof.S.B.Natikar received the M.Tech in computer science and engineering at Basaveshwara Engineering College, Bagalkot under VTU, belgaum. He is working as Assistant Professor in Vishwabharati Academy's College of Engineering, Savitribai Phule Pune University, Ahmednagar, Maharashtra, India. His research interests include networking,wireless network,cloud computing,advanced databases and mobile computing.