ISSN:0975-9646

Finding Bug by using Data Reduced Techniques

Amruta Gadekar[1] , Pranjali Taralkar [2], Nikita Waghmare[3], Rahul Dapke [4]

[1]Asst.Professor,Department of CE,Pune University,India

[2,3,4]BE.Student,Department of CE,Pune University,India

Dr.D.Y.Patil Institute of Engineering and Technology Ambi,Pune,India

Abstract— Software companies spend maximum percent of cost in negotiating with software bugs which aims to classify bugs and assign a developer to a new bug. To reduce the time cost in manual work, text classification techniques are applied to conduct automatic bug triage. In FB by using DRT, we address the problem of data reduction for bug triage, i.e.,how to reduce the scale and improve the quality of bug data. We combine instance selection with feature selection to simultaneously reduce data scale on the bug dimension and the word dimension. To determine the order of applying instance selection and feature selection, we extract attributes from historical bug data sets and build a predictive model for a new bug data set. The results show that our data reduction can effectively reduce the data scale and improve the accuracy of bug triage. Our work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

Keywords—bug repositories, bug triage, bug data reduction,feature selection, instance selection

I.INTRODUCTION

In bug repository bug is maintained as bug report which reports the texual description of reproducing the bug and updates according to status of bug fixing.Bug repository provides data platform to support many types of tasks on bugs. There are two challenges related to bug data namely the large scale and the low quality [1],[3]. On one hand, due to the daily-reported bugs, a large number of new bugs are stored in bug repositories. Taking an open source project, Eclipse, as an example, an average of 30 new bugs are reported to bug repositories per day in 2007; from 2001 to 2010, 333,371 bugs have been reported to Eclipse by over 34,917 developers and users[4].It is a challenge to manually examine such large-scale bug data in software development .

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.Bugr eportsare vital for any software development. Theyallow users to inform developers of the problems encountered while using a software.for open source large-scale software projects, the number of daily bugs is so large which makes the triaging process very difficult and challenging [2]. To avoid the bugs of a softwares, we empirically examine the results of instance selection algorithms and feature selection algorithms.

Section II describe background and Section III describes the system architecture of the proposed system. The details of instance selection, feature selection, historical data use and graph module is given in Section IV implementation and concluded in Section V.

II.BACKGROUND

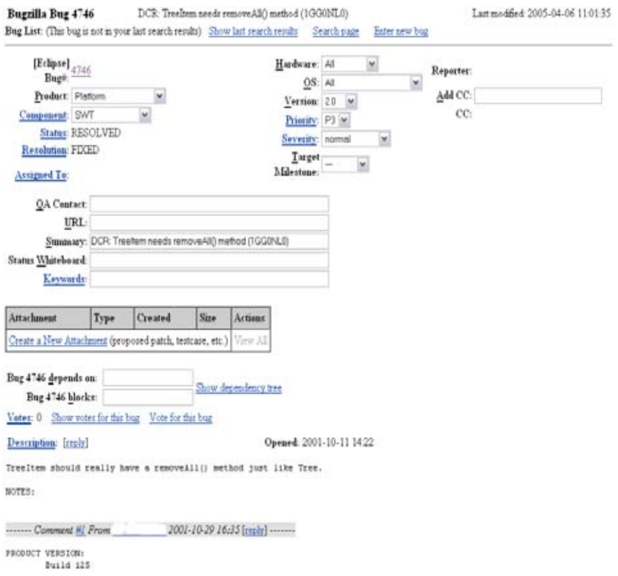
Once a software bug is found, a reporter (typically a developer, a tester, or an end user) records this bug to the bug repository. A recorded bug is called a bug report, Once a bug report is formed, a human triager assigns this bug to a developer, who will try to fix this bug. This developer is recorded in an item assigned-to. The assigned-to will change to another developer if the previously assigned developer cannot fix this bug. The process of assigning a correct developer for fixing the bug is called bug triage. A developer, who is assigned to a new bug report, starts to fix the bug based on the knowledge of historical bug fixing.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 and applying existing solutions to the new bug.Existing work employs the approaches based on text classification to assist bug triage, e.g.,[7] In such approaches,the summary and the description of a bug report are extracted as the textual content while the developer who can fix this bug is marked as the label for classification.It gives low accuracy.

A. The Lifecycle of a Bug Report

Bugs move through a series of states over their lifetime.We illustrate these states using the life-cycle of a bug report for the Eclipse bug project (Figure 1). Other projects vary slightly from this model. We describe such differences when necessary later in the paper.When a bug report is submitted to the Eclipse repository,its status is set to NEW. Once a developer has been either assigned to or accepted responsibility for the report, the status is set to ASSIGNED. When a report is closed its status is set to RESOLVED. It may further be marked as being verified (VERIFIED) or closed for good (CLOSED). A report can be resolved in a number of ways; the resolution status in the bug report is used to record how the report was resolved. If the resolution resulted in a change to the code base, the bug is resolved as FIXED. When a developer determines that the report is a duplicate of an existing report then it is marked as DUPLICATE. If the developer was unable to reproduce the bug it is indicated by setting the resolution status to WORKSFORME. If the report describes a problem that will not be fixed, or is not an actual bug, the report is marked as WONTFIX or INVALID, respectively. A formerly resolved report may be reopened at a later date, and will have its status set to REOPENED.

B. Interactions with Bug Reports

People play different roles as they interact with reports in a bug repository. The person who submits the report is the reporter or the submitter of the report. The triager is the person who decides if the report is meaningful and who assigns responsibility of the report to a developer. The one that resolves the report is the resolver. A person that configure 2: A sample Bugzilla bug report from Eclipse.Table 1: Daily bug submissions around and after product release.Around Release After Release Mean Min Max Mean Min Max Eclipse 48 1 192 13 1 124 Firefox 8 1 37 5 1 37 tributes a fix for a bug is called a contributor. A contributor may also contribute comments about how to resolve a bug or additional information that leads to the resolution of a report.A person may assume any one of these roles at any time.For example, a triager may resolve a report as the duplicate of an existing report. Alternatively, a developer may submit a report, assign it to himself, contribute a fix, and then resolve the report. For that report, a single person has fulfilled all the roles.



Bugzilla Bug 4746 DCR: TreeItem needs removeAll() method (1GG0NL0) Last modified: 2005-04-06 11:01:35

Bug List: (This bug is not in your last search results) Show last search results Search page Enter new bug

[Eclipse] Bug#:	4746	Hardware:	All	Reporter:	
Product:	Platform	OS:	All	Add CC:	
Component:	SWT	Version:	2.0	CC:	
Status:	RESOLVED	Priority:	P3		
Resolution:	FIXED	Severity:	normal		
Assigned To:		Target Milestone:	---		

QA Contact:

URL:

Summary: DCR: TreeItem needs removeAll() method (1GG0NL0)

Status Whiteboard:

Keywords:

Attachment	Type	Created	Size	Actions
Create a New Attachment (proposed patch, testcase, etc.)				View All

Bug 4746 depends on: Show dependency tree

Bug 4746 blocks:

Votes: 0 Show votes for this bug Vote for this bug

Description: [reply] Opened: 2001-10-11 14:22

```
TreeItem should really have a removeAll() method just like Tree.

NOTES:
```

------- Comment #1 From 2001-10-29 16:35 [reply] -------

```
PRODUCT VERSION:
        Build 125
```

Figure 1: A sample Bugzilla bug report from Eclipse.

III. SYSTEM ARCHITECTURE

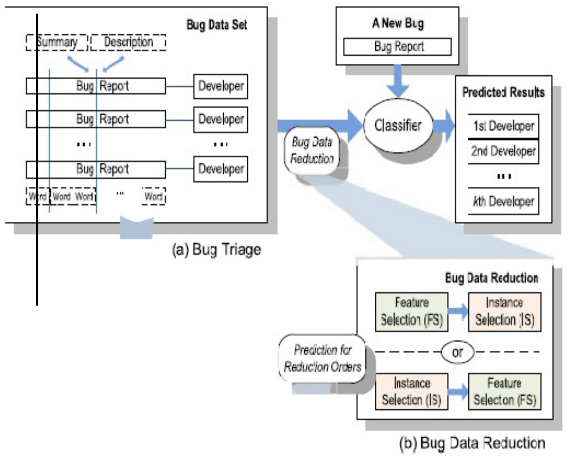


Fig. 2. Illustration of reducing bug data for bug triage is : a)Bug Triage-It describe the framework of existing work on bug triage. b)Bug Data Reduction – it combines the techniques of instance selection and featue selection to reduce scale of bug data.

A. Bug Triage

The goal of bug triage is to assign a new-coming bug to the correct potential developer. In bug triage,a bug data set is converted into a text matrix with two dimensions, namely the bug dimensions and word dimension.

B. Data Reduction for bug triage

Bug data reduction to reduce the scale and to improve the quality of data in bug repositories. Which is applied as a phase in data preparation of bug triage. We combine existing techniques of instance selection and feature selection to remove certain bug reports and words.

IV. IMPLEMENTATION

A. Instance selection

Instance selection is a technique to reduce the number of instances by removing noisy and redundant instances[5]. By using this technique original data sets are reduced by removing non-representative instances.

For a given data set in a certain application, instance selection is to obtain bug reports in bug data

B. Feature selection

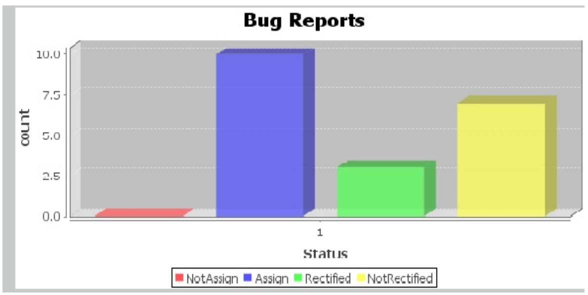
Feature selection aims to obtain a subset of relevant features (i.e., words in bug data). It is a preprocessing technique used for selecting a reduced set of features for large Scale data sets[6].

In our work we leverage the combination of Instance selection and Feature selection to generate a bug data set.

C . Graph Module

This module show's four part's as follow:

1) Firstly it will show how many bugs are not assigned to any developer. It will give complete status about the bugs to the admin so that he will come to know which bugs are not assigned yet.
2) Secondly it will show how many bugs are not assigned to any developer. It will give complete status about the bugs to the admin so that he will come to know which bugs are assigned.
3) Thirdly it will show how many bugs are rectified by the developer's. It will give complete status about the bugs to the admin so that he will come to know which bugs are rectified completely.
4) Fourthly it will show how many bugs are not rectified by the developer's. It will give complete status about the bugs to the admin so that he will come to know which bugs are not rectified yet.



V.SYSTEM ANALYSIS

1) we present the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspecs, namely

a) To simultaneously reduce the scales of the bug dimension and word dimension
b) To improve the accuracy of bug triage.

2) We propose a combination approach to addressing the problem of data reduction. This can be viewed as an application instance selection and feature selection in bug repositories.

3) We build a binary classifier to predict the order of applying instance selection and feature selection. To our knowledge the order of applying instance selection and feature selection has not been investigated in related domains.

VI. CONCLUSIONS

In this paper we have focused on minimizing bug data set in order to have less scale of data and quality data. Our work provide an approach to leverging technique to form reduced and high quality bug data in software development and maintaince. Our experimental results showed that this data reduction technique will give quality data as well as it will reduce the data scale.

In future work, we plan on improving the results of data reduction in bug triage to explore how to prepare a high quality bug data set and tackle a domain-specific software task and we want to investigate effect of other term selection methods.

REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy, "*Who should fix this bug?*" in Proc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361–370.

[2] Mamdouh Alenezi and Kenneth Magel, Shadi Banitaan "*Efficient Bug Triaging Using Text Mining*" © 2013 academy publisher

[3] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," ACMTrans. Soft. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.

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

[5] V. Cerver_on and F. J. Ferri, "Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule," IEEE Trans. Syst., Man, Cybern., Part B, Cybern., vol. 31, no. 3, pp. 408–413, Jun. 2001.

[6] 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.

[7] 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.