Bug Triage with Bug Data Reduction

Mrunal V. Wankhede
Department of Computer Engineering
BDCOE
mrunalwankhede64@gmail.com

Samiksha P. Mahakalkar
Department of Computer Engineering
BDCOE
samikshamahakalkar07@gmail.com

Shrutika P. Ingole
Department of Computer Engineering
BDCOE
shrutikaingole18@gmail.com

Snehal V. Alone
Department of Computer Engineering
BDCOE
snehalalone@gmail.com

Abstract The process of fixing bug is bug triage, which aims to correctly assign a developer to a new bug. Software companies spend most of their cost in dealing with these bugs. To reduce time and cost of bug triaging, we present an automatic approach to predict a developer with relevant experience to solve the new coming report. In proposed approach we are doing data reduction on bug data set which will reduce the scale of the data as well as increase the quality of the data. We are using instance selection and feature selection simultaneously with historical bug data. We have added a new module here which will describe the status of the bug like whether it assigned to any developer or not and it is rectified or not. Software companies spend over 45 percent of cost in dealing with software bugs. An inevitable step of fixing bugs is bug triage, which aims to correctly assign a developer to a new bug. To decrease the time cost in manual work, text classification techniques are applied to conduct automatic bug triage. In this paper, 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. Bug triage is an essential step in the process of bug fixing. Bug triage is the process of fixing bug whose main objective is to correctly allocate a developer to a new bug further handling. Many software companies spend their most of cost in dealing with these bugs. To decrease the time cost in manual work and to enhance the working of automatic bug triage, two techniques are applied namely text classification and binary classification. In literature various papers address the problem of data reduction for bug triage, i.e., how to reduce the scale and improve the quality of bug data.

Key Words: *Bug, Bug triage, data reduction, Instance selection, Data Mining.*

*****

I. 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. For open source large-scale software projects, the number of daily bugs is so large which makes the triaging process very difficult and challenging. 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 a bug repository, a bug is maintained as a bug report, which records the textual description of reproducing the bug and updates according to the status of bug fixing.

Primary contribution of this paper is as follow: Here in this paper we are using feature selection and instance selection with historical data for reducing the bug data in bug repository so that we get quality data as well as low scale data. We are also adding a graph module for representing the bug reports. Mining software repositories is an interdisciplinary domain, which aims to employ data mining to deal with software engineering problems. Traditional software analysis is not completely suitable for the large - scale and complex data in software repositories .A bug repository (a typical software repository, for storing details of bugs), plays an important role in managing software bugs.

Software bugs are inevitable and fixing bugs are expensive in software development.

Software companies spend over percent of cost in fixing bugs. Large software projects deploy bug repositories (also called bug tracking systems) to support information collection and to assist developers to handle bugs. In a bug repository, a bug is maintained as a bug report, which records the textual description of reproducing the bug and updates according to the status of bug fixing. A bug repository provides a data platform to support many types of tasks on bugs e.g., fault prediction, bug localization, and reopened bug analysis. In this paper, bug reports in a bug repository are called bug data.

II. LITERATURE SURVEY

Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weiqin Zou used to determine the order of applying instance selection and feature selection, they extract attributes from historical bug data sets and build a predictive model for a new bug data set and identified the developer prioritization which can distinguish developers and assists task in software maintenance.

Sara Anjum illustrate an automatic approach to predict a developer with relevant experience to solve the new coming report and also mentioned that Bug triaging is an error-prone,tedious and time consuming task and also focused with revisiting bug triage and resolution practice.

Mamdouh Alenezi and Kenneth Magel, Shadi Banitaan Investigate the use of five term selection methods on the accuracy of bug assignment and in this paper they studied about bug triaging and fixing practices contains with bug reassignments and reopening.

Ahmed Lamkanfi, Serge Demeyery, Emanuel Gigery, Bart Goethalsz This paper illustrates the severity of a reported bug by analyzing its textual description using text mining algorithms and the bug traiging by semi supervised approah,unlabeled bug reports were added to the existing supervised approach to getrid of the deficiency of labelled bug reports.

Shay Artzi, Adam Kie zun, Julian Dolby, Frank Tip, Danny Dig, Amit Paradkar This paper illustrates how to Find Bugs in Web Applications Using Dynamic Test Generation and the approach is introduced on a collection of 15,859 bug reports from a large scale project.

III. EXISTING SYSTEM

A time-consuming step of handling software bugs is bug triage, which Objective is to assign a correct developer to fix a new bug. In conventional software development, new bugs are manually triaged by an ex pert developer, i.e., a human triage. Due to the large n umber of daily bugs and the lack of expertise of all the bugs, manual bug triage is costly in time cost and low in accuracy. In manual bug triage in Eclipse, percent of bugs are a signed by mistake while the time cost between opening one bug and its first triaging is 19.3 days on average. To evade the expensive cost of manual bug triage, existing work has anticipated an automatic bug triage approach,

which applies text classification techniques to foretell developers for bug reports. In this approach, a bug report is mapped t o a document and a related developer is mapped to the label of the document.

Then, bug triage is changed into a problem of text classification and is automatically resolved with mature text classification techniques, e.g., Naive Bayes. Based on the results of text classification, a human triage assigns new bugs by incorporating his/her expertise. To improve the accuracy of text classification techniques for bug triage, some further techniques are investigated, e.g., a tossing graph approach and a collaborative filtering approach. Though large-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage. Since software bug data are a kind of free-form text data (generated by developers), it is required to generate well-processed bug data to make easy the application In this paper, we address the problem of data reduction for bug triage, i.e., how to diminish the bug data to save the labour cost of developers and improve the quality to facilitate the process of bug triage.

Data reduction for bug triage aims to construct 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 join existing techniques of instance selection and feature select ion 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 t he bias of a single algorithm, we empirically examine the results of four instance selection algorithms and four feature selection algorithm.

IV. PROPOSED SYSTEM

In this part, we present the data preparation for applying the bug data reduction. We assess the bug data reduction on bug repositories of two large open source projects, namely Eclipse and Mozilla. Eclipse is a multi-language software development environment, including an Integrated Development Environment (IDE) and an extensible plug-in system; Mozilla is an Internet application suite, including some classic products, such as the Firefox browser and the Thunderbird email client. Up to December 31, 2011, 366,443 bug reports over 10 years have been recorded to Eclipse while 643,615 bug reports over 12 years have been recorded to Mozilla. In our work, we gather continuous 300,000 bug reports for each project of Eclipse and Mozilla, i.e., bugs 1-300000 in Eclipse and bug s 300001600000 in Mozilla.

Actually, 298,785 bug reports in Eclipse and 281,180 bug reports in Mozilla are unruffled since some of bug reports are removed from bug repositories (e.g., bug 5315 in Eclipse) or not allowed anonymous access (e.g., bug 40020 in Mozilla). For each bug report, we download web pages from bug repositories and extract the details of bug reports for experiments. Since bug triage aims to forecast the developers who can fix the bugs, we follow the existing work to eliminate unfixed bug reports, e.g., the new bug reports or will-not-fix bug reports. Thus, we only wish bug reports, which are fixed and duplicate (based o n the items status of bug reports). In addition in bug repositories, several developers have only fixed very few bugs. Such inactive developers may not provide sufficient information for fire casting correct developers. In our work, we remove the developers, who have fixed less than 10 bugs.

A. Word Dimension

We use feature selection to eliminate noisy duplicate words in a data set.

B. Bug dimension

Instance selection can remove uninformative bug reports; mean while, we can observe that the accuracy may be decreased by removing bug reports.

C. *Word dimension*

By eliminating the uninformative words, feature selection improves the accuracy of bug triage. This can recover the accuracy loss by instance selection.

V. ARCHITECTURE

Aim of bug triage is to assign a developer for bug fixing once a developer is assigned to a new bug report he will fix the bug or try to rectify it. In bug repositories, all the bug reports are filled by developers in natural languages. The low-quality bugs accumulate in bug repositories with the growth in scale.

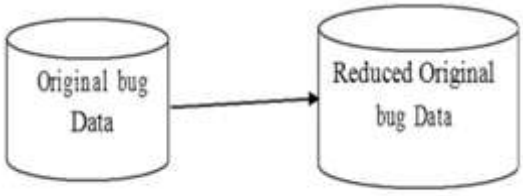

Fig. 5.1. Reduced Data Set

1) Data Reduction Here we are reducing the bug data by using instance and feature se lection so that we get low scale as well as quality data.
2) Classifier In this section, original data set is classified with the upcoming new bug report. Here we are reducing the bug data by using instance and feature selection so that we get low scale as well as quality data in solving bug report.

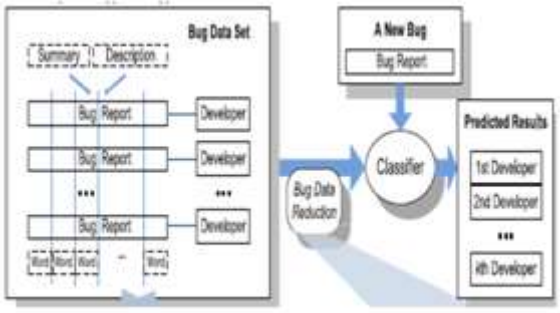

Fig. 5.2. Process of Bug Triage

VI. GRAPH MODULE

This module shows four parts 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 developers. 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 developers. It will give complete status about the bugs to the admin so that he will come to know which bugs are not rectified yet.

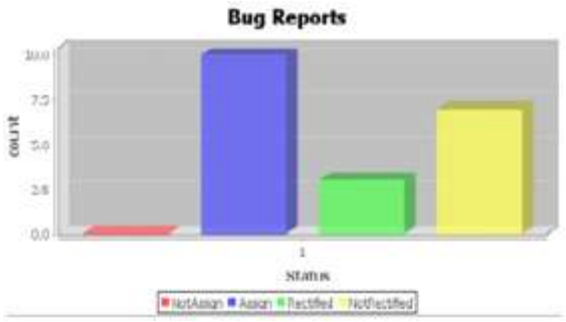

Fig. 6.1. Graph Module of Bug Triage

VII. DATA REDUCTION FOR BUG TRIAGE

A. Applying Instance Selection and Feature 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 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) while feature selection aims to obtain a subset of relevant features (i.e. words in bug data). 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..

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. Two algorithms FS and IS are applied sequentially. Note that in Step, some of bug reports may be blank during feature selection, i.e., all the words in a bug report are removed. Such blank bug reports are also removed in the feature selection. In our work, FS ! IS and IS ! FS are viewed as two orders of bug data reduction.

Feature selection is a pre-processing technique for selecting a reduced set of features for large scale data sets. The reduced set is considered as the representative features of the original feature set. Since bug triage is converted into text classification, we focus on the feature selection algorithms in text data. In this paper, we choose four well performed algorithms in text data and software data, namely Information Gain (IG), x2 statistic (CH) , Symmetrical Uncertainty attribute evaluation(SU) ,and Relief-F Attribute selection (RF). Based on feature selection, words in bug reports are sorted according to their feature values and a given number of words with large values are selected as representative features.

VIII. MERITS AND DEMERITS

A. Merits:

1) Reducing the data scale
2) Improving the accuracy of bug triage
3) It save the labour cost of developers

B. Demerits:

1) Traditional bug triage system is cost effective process.
2) Also bug type is critical means consumes lot of stages also coinsured with time.

IX. CONCLUSION

Bug triage is an expensive step of software maintenance in both labour cost and time cost. In this paper, we combine feature selection with instance selection to reduce the scale of bug data sets as well as improve the data quality. To deter- mine the order of applying instance selection and feature selection for a new bug data set, we extract attributes of each bug data set and train a predictive model based on historical data sets. We empirically investigate the data reduction for bug triage in bug repositories of two large open source projects, namely Eclipse and Mozilla. Our work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

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. For predicting reduction orders, we plan to pay efforts to find out the potential relationship between the attributes of bug data sets and the reduction orders.

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] Mamdouh Alenezi and Kenneth Magel, Shadi Banitaan Efficient Bug Triaging Using Text Mining 2013 academy publisher

[3] Sara Anjum, Pankaj Gakare,Yogita Dhole Bug Triage with Bug Data Reduction Volume: 02 Issue: 04 July-2015

[4] Shay Artzi, Adam Kiezun, Julian Dolby, Frank Tip, Danny Dig, Amit Paradkar Finding Bugs inWeb Applications Using Dynamic Test Generation and Explicit State Model Checking VOL. 36, NO. 4, JULY/AUGUST 2010

[5] Ahmed Lamkanfi, Serge Demeyery, Emanuel Gigery, Bart Goethalsz Predicting the Severity of a Reported Bug 2010 IEEE