RESEARCH ARTICLE OPEN ACCESS

Towards Effective Bug Triage with Software Data Reduction Techniques

G.Loganayagi, A.Petrisia
[1]PG Student, [2]Associate Professor

Department of MCA, Dhanalakshmi Srinivasan College of Engineering and Technology

Abstract:

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. 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. We empirically investigate the performance of data reduction on totally 600,000 bug reports of two large open source projects, namely Eclipse and Mozilla. 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.

INTRODUCTION

Mining software repositories is an inter disciplinary domain, which aims to employ data mining to deal with software engineering problems. 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. Traditional software analysis is not completely suitable for the large-scale and complex data in software repositories .Data mining has emerged as a promising means to handle software data . 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. Software bugs are inevitable and fixing bugs is expensive in software development. Software companies spend over 45 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 EXISTING SYSTEM

A time-consuming step of handling software bugs is bug triage, which aims to assign a correct developer to fix a new bug. 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. In manual bug triage in Eclipse, percent of bugs are assigned by mistake while the time cost between opening one bug and its first triaging is 19.3 days on average. To avoid the expensive cost of manual bug triage, existing work has proposed an automatic bug triage approach, which applies text classification techniques to predict developers for bug reports. In this approach, a bug report is mapped to a

document and a related developer is mapped to the label of the document. Then, bug triage is converted into a problem of text classification and is automatically solved with mature text classification techniques, e.g., Naive Bayes. Based on the results of text classification, a human triager 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. However, 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, it is necessary to generate well-processed bug data to facilitate the application.

III PROPOSED SYSTEM

Since bug triage aims to predict the developers who can fix the bugs, we follow the existing work to remove unfixed bug reports, e.g., the new bug reports or will-not-fix bug reports. Thus, we only choose bug reports, which are fixed and duplicate (based on the items status of bug reports). Moreover, in bug repositories, several developers have only fixed very few bugs. Such inactive developers may not provide sufficient information for predicting correct developers. In our work, we remove the developers, who have fixed less than 10 bugs.

IV.SYSTEM DESCRIPTION

It includes modules like Data reduction,Data reduction for bug triage,Applying Instance Selection and Feature Selection,Reduction Orders.

A. Data reduction

Data reduction is the transformation of numerical or alphabetical digital information derived empirically or experimentally into a corrected, ordered, and simplified form. The basic concept is the reduction of multitudinous amounts of data down to the meaningful parts. When information is derived from instrument readings there may also be a transformation from analog to digital form. When the data are already in digital form the 'reduction' of the data typically involves some editing, scaling, coding, sorting, collating, and producing tabular summaries. When the observations are discrete but the underlying phenomenon is continuous then smoothing and interpolation are often needed. Often the data reduction is undertaken in the presence of reading or measurement errors. Some idea of the nature of these errors is needed before the most likely value may be determined.

B. Data reduction for bug triage

We propose bug data reduction to reduce the scale and to improve the quality of data in bug repositories. We combine existing techniques of instance selection and feature selection to remove certain bug reports and words. A problem for reducing the bug data is to determine the order of applying instance selection and feature selection, which is denoted as the prediction of reduction orders. In this section, we first present how to apply instance selection and feature selection to bug data, i.e., data reduction for bug triage. Then, we list the benefit of the data reduction.

C. 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 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) 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 2), 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.

D. Reduction Orders

To apply the data reduction to each new bug data set, we need to check the accuracy of both two orders (FS ! IS and IS!FS) and choose a better one. To avoid the time cost of manually checking both reduction orders, we consider predicting the reduction order for a new bug data set based on historical data sets. We convert the problem of prediction for reduction orders into a binary classification problem. A bug data set is mapped to an instance and the associated reduction order (either FS ! IS or IS ! FS) is mapped to the label of a class of instances. Note that a classifier can be trained only once when facing many new bug data sets. That is, training such a classifier once can predict the reduction orders for all the new data sets without checking both reduction orders. To date, the problem of predicting reduction orders of applying feature selection and instance selection has not been investigated in other application scenarios. From the perspective of software engineering, predicting the reduction order for bug data sets can be viewed as a kind of software metrics, which involves activities for measuring some property for a piece of software. However, the features in our work are extracted from the bug data set while the features in existing work on software metrics are for individual software artifacts,3 e.g., an individual bug report or an individual piece of code. In this paper, to avoid ambiguous denotations, an attribute refers to an extracted feature of a bug data set while a feature refers to a word of a bug report.

E. Implementation

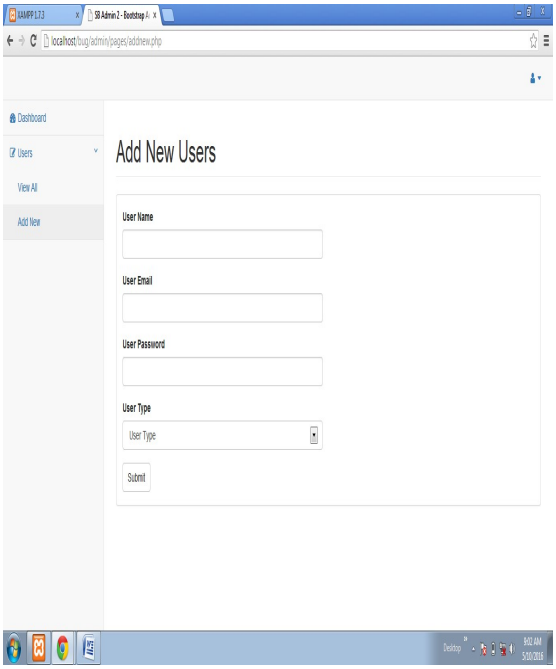

Fig 1

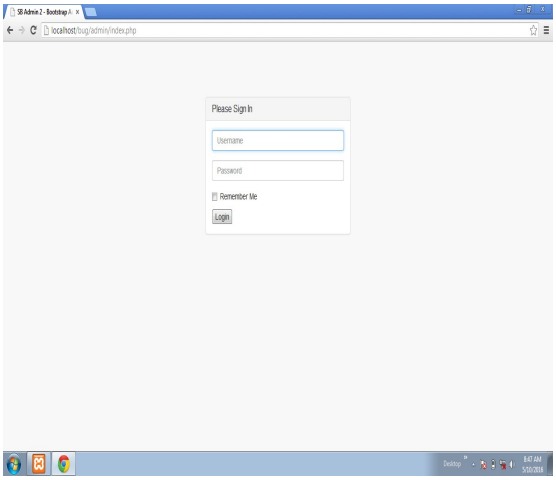

Fig 2

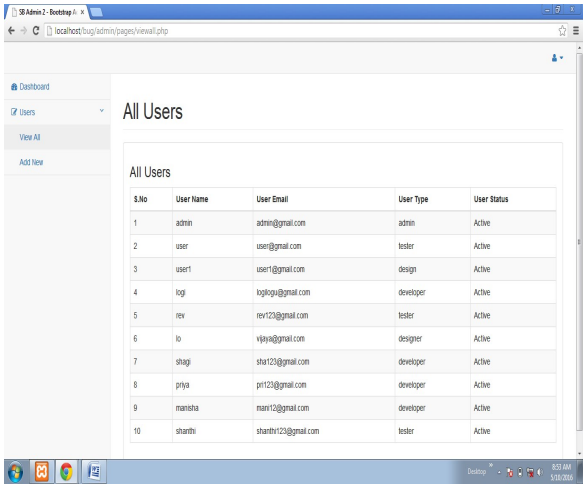

Fig 3

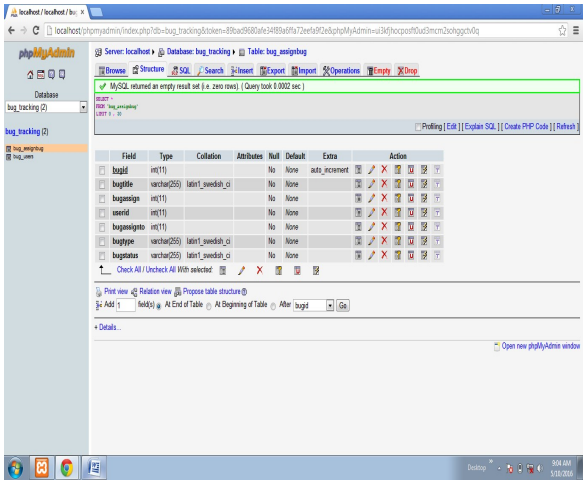

Fig 4

V.CONCLUSION

Bug triage is an expensive step of software maintenance in both labor 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 determine 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.

VI.REFERENCE

1. Advanced PHP for Flash by Steve Webster, et al .friends of ED. Paperback- September 2002.
2. Advanced PHP for Web Development (The Prentice Hall PTR Advanced Web Development Series) by Christopher Cosentino.Prentice Hall PTR. Paperback- 1 October, 2002.
3. Advanced PHP Programming by Schlossnagle.Sams. Paperback- October 2003.
4. A Programmer's Introduction to PHP by W.J. Gilmore.Apress. Paperback- 1 January, 2001
5. Beginning PHP 4 Databases by Deepak Thomas, et al .Wrox Press Ltd. Paperback- 17 October, 2002.
6. Beginning PHP4 Programming by John Blank, et al .Wrox Press Ltd. Paperback- 30 October, 2000
7. Beginning PHP, MySQL and Apache. Wrox Press Ltd. Paperback- 1 June, 2003.
8. Building a PHP Intranet Problem Design Solution by Wrox Author Team.WROXP.. Paperback- 31 December, 2004.
9. Building Database Applications on the Web Using PHP3 by Craig Hilton, Jeff Willis.Addison Wesley. Paperback- December 1999
10. Core PHP Programming by Leon Atkinson.Prentice Hall PTR. Paperback- 1 August, 2000