International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

A Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage

Suvarna Kale[1], **Ajay Kumar Gupta**[2]

[1]M.E (Computer), Dept. Of Computer Engg., SP's IOK College of Engineering, Savitribai Phule Pune University, Pimpale Jagtap, Pune, Maharashtra, India

[2]Assistant Professor, Dept. Of Computer Engg., SP's IOK College of Engineering, Savitribai Phule Pune University, Pimpale Jagtap, Pune, Maharashtra, India

Abstract: ***Software organizations spend more than 45 percent of expense in managing Software bugs. An unavoidable step of altering bugs is bug triage, which plans to accurately allot a developer to another bug. To reduce the time cost in manual work, text classification methods are used for automatic bug triage. In this undertaking, we address the issue of data reduction for bug triage, i.e., how to diminish the scale and improve the nature of bug data. We consolidate instance selection with feature selection to simultaneously decrease information scale on the bug dimension and the word dimension. This paper gives a way to deal with utilizing methods on data processing to form reduced and high-quality bug data in software development and maintenance and also improve the results of data reduction in bug triage.***

Keywords: bug data reduction, feature selection, instance selection, bug triage.

1. Introduction

In the software engineering mining software repositories is an interdisciplinary area, which means to utilize idata mining to deal with software engineering problems. In modern software development, software repositories are huge scale databases for putting the output of software development, e.g., source code, bugs, and so forth. Conventional software analysis is not totally suitable for the large-scale and complex data in software repositories. Data mining has developed as a promising means to handle software data. By utilizing 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), assumes a vital part in managing software bugs. Software bugs are inescapable and fixing bugs is costly in software development. Software companies spend more than 45 percent of expense in fixing bugs. Large software projects convey bug repositories (also called bug tracking systems) support information collection and to assist developers to handle bugs. In a bug repository, a bug is kept as a bug report, which records the textual description of reproducing the bug and upgrades as per the status of bug fixing. A bug repository gives an data stage upport 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.

There are two difficulties related to bug data that may influence the effective use of bug repositories in software development tasks, namely the huge scale and the low quality. On one hand, because of the every day reported bugs, a large number of new bugs are stored in bug repositories. Taking an open source projects, Eclipse, as an example, a normal of 30 new bugs are accounted for to bug repositories every day in 2007; from 2001 to 2010, 333,371 bugs have been accounted for to Eclipse by more than 34,917 developers and users. It is challenge to manually look at such expansive scale bug data in software development. Then again, software techniques suffer from the low quality of bug data. Two regular attributes of low-quality bugs are noise and redundancy. Noisy bugs may delude related developers while excess bugs waste the limited time of bug handling.

1.1 Problem Statement

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.

2. Literature Survey

J. Anvik, L. Hiew, and G. C. Murphy present a semi-automated approach intended to ease one part of this process, the assignment of reports to a developer. S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D. Ernst presents Apollo's algorithms and implementation, and an experimental evaluation . Apollo generates test inputs for a Web application, monitors the application for crashes, and validates that the output conforms to the HTML specification. J. Anvik and G. C. Murphy shows that recommenders for which developer should fix a bug can be quickly configured with this approach and that the configured recommenders are within 15% precision of hand-tuned developer recommenders. Shivkumar Shivaj, E. James, Ram Akella, Sunghun Kim investigates multiple feature selection techniques that are generally applicable to classification-based bug prediction methods. The techniques discard less important features until optimal classification performance is reached. Ahmed Lamkanfi, Serge Demeyer, Emanuel Giger, Bart Goethals investigate whether we can accurately predict the severity of a reported bug by analyzing its textual description using text mining algorithms. Weiqin Zou ,Yan Hu, Jifeng Xuan, He Jiang propose the training set reduction with both feature selection and instance selection techniques for bug triage. We combine feature selection with instance selection to improve the accuracy of bug triage.

3. Existing System

A time-consuming step of taking care of software bugs is bug triage, which plans to assign a right developer to alter another bug. In traditional software development, new bugs are physically triaged by an expert developer, i.e., a human triager. Because of the huge number of every day bugs and the absence of skill of the considerable number of bugs, manual bug triage is costly in time cost and low in precision. In manual bug triage in Eclipse, 44 percent of bugs are assigned by mistake while the time expense between opening one bug and its first triaging is 19.3 days overall. To avoid the expensive cost of manual bug triage, existing work has proposed a automatic bug triage approach, which applies text classification techniques to predict developers for bug reports. In this methodology, a bug report is mapped to a record and a related engineer is mapped to the name of the archive. At that point, bug triage is changed over into an issue of content grouping and is consequently tackled with experienced content characterization systems.

Disadvantages

It presents the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspects, namely

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

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

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.

4. Proposed System

In this part, we present the data preparation for applying the bug data reduction. We evaluate 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 collect continuous 300,000 bug reports for each project of Eclipse and Mozilla, i.e., bugs 1-300000 in Eclipse and bugs 300001600000 in Mozilla. Actually, 298,785 bug reports in Eclipse and 281,180 bug reports in Mozilla are collected 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 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.

Advantages:
Word dimension: We use feature selection to remove noisy duplicate words in a data set. By removing uninformative words, feature selection improves the accuracy of bug triage. This can recover the accuracy loss by instance selection.

Bug dimension: Instance selection can remove uninformative bug reports, meanwhile, we can observe that the accuracy may be decreased by removing bug reports.

5. Motivation

Real-world data always include noise and redundancy. Noisy data may misdirect the data examination procedures while redundant data may expand the expense of data processing. In bug repositories, all the bug reports are filled by developers in natural languages. The low-quality bugs aggregate in bug repositories with the development in scale. Such expansive scale and low-quality bug information may break down the effectiveness of fixing bugs. So in this work we will demonstrate the need for data reduction.

Instance selection and feature selection are broadly utilized systems in information handling. For a given information set in a certain application, instance selection is to get a subset of applicable examples (i.e., bug reports in bug information) while feature selection expects to get a subset of significant elements (i.e., words in bug information). In our work, we utilize the mix of instance selection and feature selection. To recognize the requests of applying instance selection and feature selection, we give the accompanying signification. Given an instance selection algorithm IS and an feature selection algorithm FS, we utilize FS! IS to signify the bug data reduction, which first applies FS and after that IS; then again, IS! FS signifies first applying IS and after that FS.

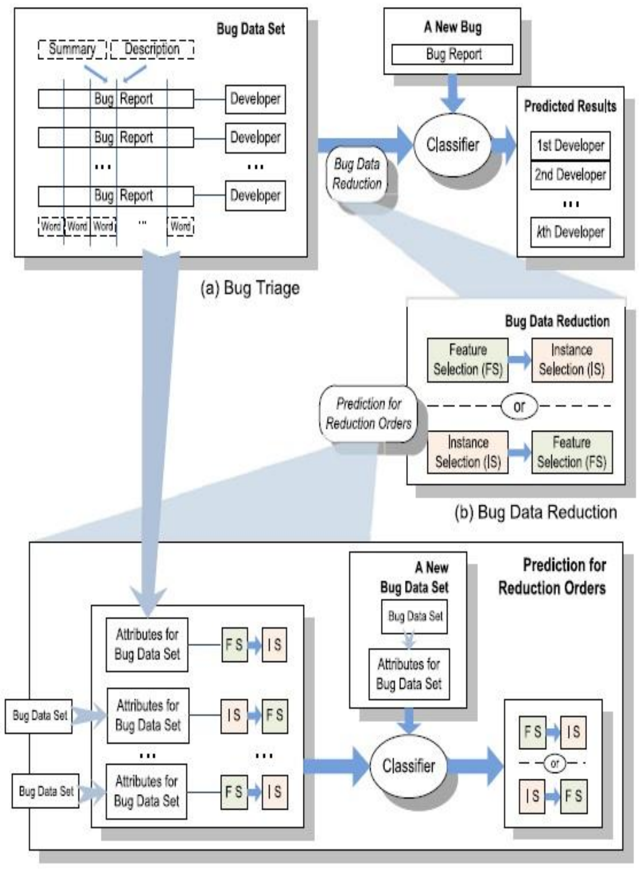


Figure 1: Illustration of reducing bug data for bug triage. Sub-figure (a) presents the framework of existing work on bug triage. Before training a classifier with a bug data set, we add a phase of data reduction, in (b), which combines the techniques of instance selection and feature selection to reduce the scale of bug data. In bug data reduction, a problem is how to determine the order of two reduction techniques. In (c), based on the attributes of historical bug data sets, we propose a binary classification method to predict reduction orders.

6. System Architecture

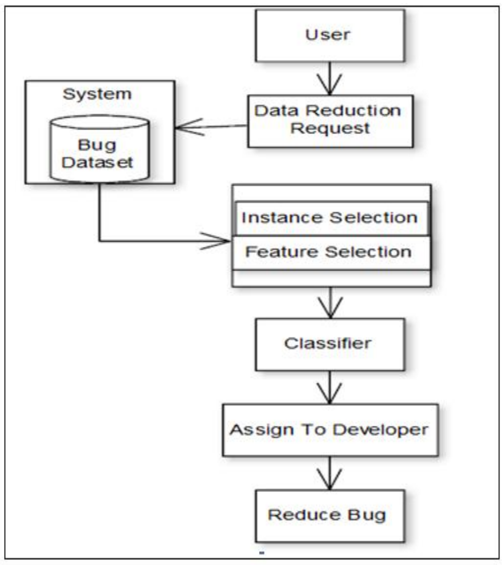


Figure 2: Architecture of System

Bug repositories are broadly utilized for keeping up software bugs, e.g., a popular and open source bug repository, Bugzilla. Once a software bug is found, a journalist (normally an engineer, an tester, or an end client) records this bug to the bug repository. A recorded bug is known as a bug report, which has numerous things for specifying the information of reproducing the bug. In a bug report, the summary and the description are two key things about the data of the bug, which are recorded in natural languages. As their names suggest, the summary signifies a general statement for identifying a bug while the description gives the details for reproducing the bug. When 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 procedure of allotting a right developer for fixing the bug is called bug triage. A developer, who is appointed to new bug report, begins to fix the bug taking into account the knowledge of historical bug fixing. 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. 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.

7. Module Description

Instance Selection:
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.

Data Reduction:
In our work, to save the labor cost of developers, the data reduction for bug triage has two goals.
1) Reducing the data scale.
2) Improving the accuracy of bug triage.

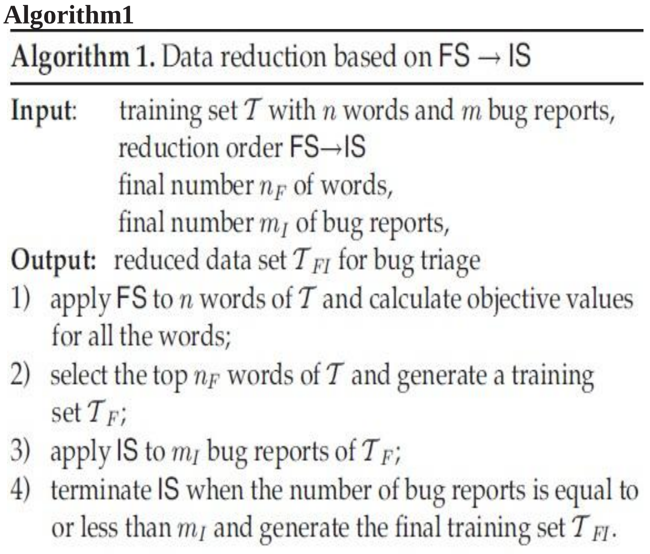
In contrast to modeling the textual content of bug reports in existing work, we aim to augment the data set to build a preprocessing approach, which can be applied before an existing bug triage approach.

Mathematical Model

INPUT:-
Let S is the Whole System Consist of
S= {U, FS, IS, ICF, DROP, POP, O}
Where,
IS = instance selection.
FS = feature selection.
ICF = Iterative Case Filter.
DROP = Decremental Reduction Optimization Procedure.
POP = Patterns by Ordered Projections.

Procedure:
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.

Algorithm1



```
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 $T_{FI}$.
```

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)

OUTPUT: Getting the appropriate Decision from the above technique

8. Conclusion and Future Scope

Bug triage is a costly stride of software maintenance in both work cost and time cost. In this paper, we combine feature selection with instance selection to diminish the size of bug information sets and additionally enhances the information quality. To decide the request of applying instance selection and feature selection for another bug data set, we extract attributes of each bug data set and train a predictive model based on historical data sets. Our work gives a way to deal with utilizing strategies on data processing to form reduced and high-quality bug data in software development and maintenance. 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] 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] S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D.Ernst, “Findingbugs in web applications using dynamic test generationand explicit-state model checkin,” IEEE Softw., vol. 36,no. 4, pp. 474–494, Jul./Aug. 2010.
[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] C. C. Aggarwal and P. Zhao, “Towards graphical models for textprocessing,” Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.
[5] K. Balog, L. Azzopardi, and M. de Rijke, “Formal models forexpert finding in enterprise corpora,” in Proc. 29th Annu. Int. ACMSIGIR Conf. Res. Develop. Inform. Retrieval, Aug. 2006, pp. 43–50.

Author Profile

Miss. Suvarna Kale received the B.E. (Computer Engineering) from University of Pune in 2011 and pursuing M.E. degree in Computer Engineering from Institute of Knowledge College of Engineering, Savitribai Phule Pune University, Pimpale Jagtap Pune, Maharashtra, India in 2015-16.

Mr. Ajay Kumar Gupta is working as Assistant Professor in Institute of Knowledge of COE, Savitribai Phule Pune University, Pimpale Jagtap, Pune, Maharashtra, India.