International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 03 | Mar-2016 www.irjet.net p-ISSN: 2395-0072

IMPROVED APPROACH FOR PREDICTING THE BUG TRIAGE USING DATA REDUCTION METHODS

S. Suganya [1], R. Venkadesh [2]

[1]*PG Scholar, Department of Computer Science and Engineering, Mahendra Engineering College, Namakkal, India*
Sugu.charming@gmail.com

[2]*Assistant Professor, Department of Computer Science and Engineering, Mahendra Engineering College, Namakkal, India*
venkadeshr@mahendra.info

Abstract - *Most of the software companies need to deal with vast number of software bugs day to day. This paper can be viewed as an application of instance selection and feature selection in bug repositories. The aim is to address the problem of data reduction for bug triage, and to reduce the scale and improve the quality of bug data. This can be achieved by combining instance selection with feature selection to simultaneously reduce data scale on the bug dimension and the word dimension. Data reduction can be addressed by combining instance selection and feature selection approaches.The reduced data set contains lesser bug data with fewer bug reports and fewer words than the original bug data and provide similar information over the original bug data and build a binary classifier to predict the order of applying instance selection and feature selection. Determine the order of applying instance selection and feature selection, to extract attributes from historical bug data sets and build a predictive model for a new bug data set. Empirically investigate the performance of data reduction on bug reports of two large open source projects, namely Eclipse and Mozilla.*

Key Words*: *Bug, Bug data reduction, Feature Selection, Instance Selection, Bug Triage, Data Mining

1. INTRODUCTION

Data mining, or knowledge discovery, is the computer-assisted process of scrutinizing and analyzing enormous sets of data and then extracting the knowledge from the data. The tools that focus in the Data mining process are called Data mining tools. These tools can be employed to visualize behaviors and future trends, allowing businesses to take proactive, knowledge-driven decisions. Data mining tools can answer business questions intelligently that traditionally were time consuming to resolve. They examine the databases for hidden patterns, finding predictive information that experts may miss because it may lies outside their expectations. The gradual increase in the amount of digital stored data was further enhanced by the success of the relational model for storing data and the development and maturing of data retrieval and manipulation technologies [2]. While technology for storing the data was developed swiftly to keep up with the demand, very little attention was paid to develop software for analyzing the data, until recently when companies realized that hidden within these masses of data was a resource that was being ignored. The huge amounts of stored data contain knowledge about a number of aspects of their business waiting to be harnessed and utilized for more effective business decision support. Database Management Systems, which are used to manage these data's at present, allow the user to retrieve only the information present explicitly in the databases. The data mining process is to extract information from the data base data and remodel it into an understandable structure for further decision making, a suitable example is the Stock market data. This extraction of knowledge from large data sets is called Data mining or Knowledge Discovery in Databases and is defined as the non-trivial extraction of implicit, previously unknown and potentially useful information from database data. A bug tracking system or defect tracking system is a software application that keeps track of reported software bugs in software development projects for further analysis [3]. It can also be referred to as issue tracking system. Many bug tracking systems, such as those used by most open source software projects, allow end-users to enter bug reports directly. Other systems are used only internally in a company or organization for doing software development. Typically bug tracking systems are integrated with other software project management applications. A bug tracking system is usually a necessary component of a good software development infrastructure, and continuous use of a bug or issue tracking is considered

as one of the "hallmarks of a good software team". Software bugs can be managed effectively using Bug repositories. 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 humongous which makes the triaging process very tedious and challenging. The process of fixing bugs is of utmost importance. This theory is supported by the fact that the majority of the software companies spend about 40 percent of their estimated expenditure in tracking down and eliminating bugs. The two factors that serve as a hindrance to the use of bug repositories in software development tasks were large scale and low quality. In a bug repository, a bug is maintained as a bug report [3], which records the textual description on the cause of the bug and also updates according to the status of bug to be fixed.

A major component of a bug tracking system is a database that maintains facts about known bugs. Facts may include the time a bug was reported, its severity, the erroneous program behavior, and details on how to reproduce the bug; as well as the identity of the person who reported it and any programmers who may be working on fixing it.

Typical bug tracking systems support the concept of the life cycle for a bug which is tracked through the status assigned to the bug. A bug tracking system should allow administrators to configure permissions based on status, move the bug to another status, or delete the bug. The system should also allow administrators to configure the bug statuses and to what extent a bug in a particular status can be moved. Some systems will e-mail interested parties, such as the bug reporter and assigned programmers, when new records are added or the status changes.

In the experiments, evaluation on the reduction of data for bug triage on bug reports were applied over two large open source projects, namely Eclipse and Mozilla. Experimental results show that applying the instance selection technique to the data set can reduce bug reports but the accuracy of bug triage may be decreased; applying the feature selection technique can reduce words in the bug data and the accuracy can be increased. Thus if both the preprocessing techniques are combined, can increase in the accuracy, as well as reduction in the size of the bug reports and words can be easily obtained. For example, when 60 percent of bugs and 80 percent of words are removed, the accuracy of Naive Bayes on Eclipse improves by 3 to 15 percent and the accuracy on Mozilla improves by 2 to 7 percent. Based on the attributes from historical bug data sets, constructed predictive model can provide accuracy of 80.8 percent for predicting the reduction order. Based on top node analysis of the attributes, results show that all attributes are essential for prediction and an individual attribute cannot predict this without other attributes [9].

Bug triage is an age-old technique that is used to handle the software bugs, whose utilization can often be time consuming. Bug triage 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 reported bugs and the lack of knowledge of all the bugs, manual bug triage is expensive in time cost and also results in low accuracy [4]. 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 such expensive cost of manual bug triage, proposals have been made to automate the bug triage approach. In this method, text classification techniques are applied to predict developers for bug reports. In this approach, a bug report is mapped to a document and a related developer is mapped as 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 may assign new bugs by incorporating their expertise [7]. The accuracy of text classification

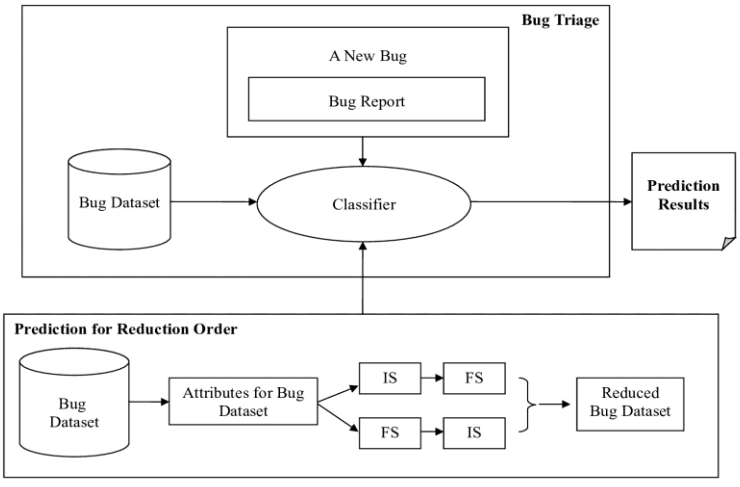

Figure : 1 Architecture of Bug Triage

techniques for bug triage can further be improved. Some techniques are: a tossing graph approach and a collaborative filtering approach [1]. However, large-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage [4]. Since software bug data are a kind of free-form text data (generated by developers), it is necessary to generate well-processed bug data to serve the application. To address the problem of data reduction for bug triage, i.e., how to reduce the bug data to save the labour cost of developers and improve the quality to facilitate the process of bug triage. Pre –processing the text data is to be done. A small-scale and high-quality set of bug data can be obtained by removing bug reports and words, which are redundant or non-informative. This is called the Data reduction in Bug triage. In this work, existing techniques of instance selection and feature selection can be combined simultaneously to reduce the bug 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. The reduced bug data is evaluated according to two criteria: the scale of a data set and the accuracy of bug triage. To avoid the bias of a single algorithm, the results of four instance selection algorithms and four feature selection algorithms were deployed to empirically examine the bug reports.

Disadvantages

Data reduction in bug triages is a tenacious process. This problem aims to augment the data set of bug triage in two aspects. They are as follows: Simultaneously reducing the scales of the bug dimension and the word dimension, to improve the accuracy of bug triage, which can be viewed as an application of instance selection and feature selection in bug repositories. Secondly,building a binary classifier to predict the order of applying instance selection and feature selection. The order of applying instance selection and feature selection has not been investigated in related domains.

Enhanced Instance Selection Algorithm

Instance selection and feature selection are widely used techniques as data processing. For a given data set in a certain application, instance selection is used to obtain a subset of relevant instances, while feature selection aims to obtain a subset of relevant features.
The quality of bug triage can be measured with the accuracy of bug triage. To avoid the bias from a single algorithm, results of four typical algorithms of instance selection and feature selection, are examined.

Bug Repository

A bug repository is a typical software repository which is used to store the details of bugs, e.g., Bugzilla. Large software projects deploy bug repositories, which are used to support information collection and to assist developers or end users to handle bugs. Each 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. The use of bug repository can improve the development process and quality of software produced. It provides a data platform to support many types of tasks on bugs, e.g., fault prediction, bug localization and reopened bug analysis [9].
A bug repository is a typical software repository, for storing details of bugs. Large software projects that deploy bug repositories can also be called as bug tracking systems, which is used to support information collection and to assist developers or end users to handle bugs. 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. It provides a data platform to support many types of tasks on bugs, e.g., fault prediction, bug localization and reopened bug analysis. A bug report is also called as bug data [9]. Software bugs are inevitable and fixing bugs is expensive in software development [10]. Software companies spend over 45 percent of cost in fixing bugs. Large software projects deploy bug repositories to support information collection and to assist developers to handle bugs. 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 [8]. There are two challenges related to bug data that may affect the effective use of bug repositories in software development tasks, namely the huge quantity and the decreased availability of data that makes actual sense. On one hand, due to the daily-reported bugs, a large number of new bugs are stored in bug repositories.
The process of assigning a correct developer for fixing the bug is called bug triage. Once the bug report is formed, a bug triager assigns the bug to a developer or stakeholder who can fix this bug and developer or end user is recorded in an item assigned-to without any tossing.

Bug Data sets Bug reports

Bug report has multiple items for detailing the information of reproducing the bug. In a bug report, the

summary and the description are two key items about the information of the bug, which are recorded in natural languages. A general statement for identifying a bug is given by the summary while the description gives the details for reproducing the bug. 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. 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 [1].

Bug Data Reduction for Bug Triage

The process of assigning a correct developer for fixing the bug is one of the important considerations of bug triage. Once the bug report is formed, a bug triager assigns the bug to a developer who can fix this bug and developer is recorded in an item assigned-to without any tossing.

Initially, bug reports needs to be preprocessed to reduce the size of the data set. By combining existing techniques of instance selection and feature selection certain bug reports and words can be removed. A problem for reducing the bug data is to determine the order of applying instance selection and feature selection [5], which is denoted as the prediction of reduction orders, A new and reduced data set is obtained from the given bug data set. Two algorithms FS and IS are applied sequentially. FS→ Feature selection is a pre-processing technique for selecting a reduced set of features (i.e., words in bug report) for large scale data sets. IS→ Instance selection is a technique to reduce the number of instances by removing noisy and redundant instances. Predictive model is used to choose one between FS→IS and IS→FS.

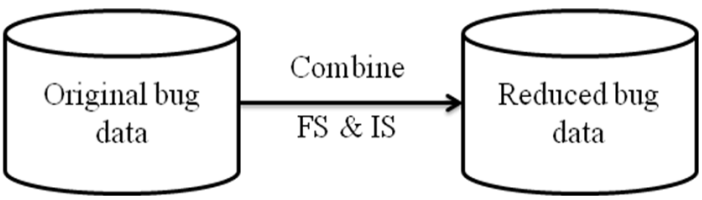

Figure : 2 Combine FS & IS

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} .

New Bug Data set

The reduced bug data set contains fewer bug reports and words than the original bug data and provides similar information over the original bug data. To apply the data reduction to each new bug data set, 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, consider predicting the reduction order for a new bug data set based on historical data sets. To avoid the time cost of manually checking both reduction orders, predicting the reduction order for a new bug data set is considered based on historical data sets.

Classifier (Text Classification Techniques)

A classifier can be applied to the new bug data sets. The input of classifier is the summary and the description is converted into the vector space model. Two steps to form the word vector space, namely tokenization and stop word removal. First, tokenize the summary and the description of bug reports into word vectors.

The text classification techniques are used to predict the developers for bug reports are,

- Extracting attributes for historical bug data sets and training a classifier.
- Predicting the reduction order for a new bug data set.
- Applying the predicted reduction order to the new bug data set.
- Triaging bug reports on the reduced data set.

A classifier can be trained only once with training (reduced) data set in order to face many new bug data set i.e., training such a classifier once can predict the reduction orders for all the new data sets without checking both reduction orders. In proposed system,

Naive Bayes classifier is used to predict the developers for new bug reports. Naïve Bayes is based on Bayesian classification Performs probabilistic prediction, i.e., predicts class membership probabilities.

Instance Selection and Feature Selection

Instance selection is a technique to reduce the number of instances by removing noisy and redundant instances. An instance selection algorithm can provide a reduced data set by removing non-representative instances. In proposed system, Iterative Case Filter (ICF) is used to reduce the bug reports. ICF defines local set L(X) which contains all cases inside largest hypersphere centred in X such that the hypersphere contains only cases of the same class a instance X. The properties of ICF are reachability and coverage is used to remove the noisy and redundant instances.

Feature selection is a pre-processing technique for selecting a reduced set of features for large-scale data sets. The pre processing techniques are tokenization, stop word removal, stemming process and vector space model. In proposed system, $\chi 2$ statistic (CH) is used to reduce the bug word dimension. The chi-squared distribution also known as chi-square or χ^2 distribution with k degrees of freedom is the distribution of a sum of the squares of k independent standard normal random variables. 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.Instance selection and feature selection are widely used techniques in data processing. For a given data set, instance selection is used to obtain a subset of relevant instances while feature selection aims to obtain a subset of relevant features. In this work, the combination of instance selection and feature selection were used to distinguish the orders of applying instance selection and feature selection, to 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.

2. Experiments on Bug Data Reduction

Data Sets and Evaluation

To examine the results of bug data reduction on bug repositories of two projects Eclipse and Mozilla. For each project, the evaluate results on five data sets and each data set is over 38 bug reports, which are fixed or duplicate bug reports [6]. We check bug reports in the two projects and find out that 40 percent of bug reports in Eclipse and 28.23 percent of bug reports in Mozilla are fixed or duplicate. Thus, to obtain over 50 fixed or duplicate bug reports, each data set in Eclipse is collected from continuous 35 bug reports while each bug set in Mozilla is collected from continuous 32 bug reports.

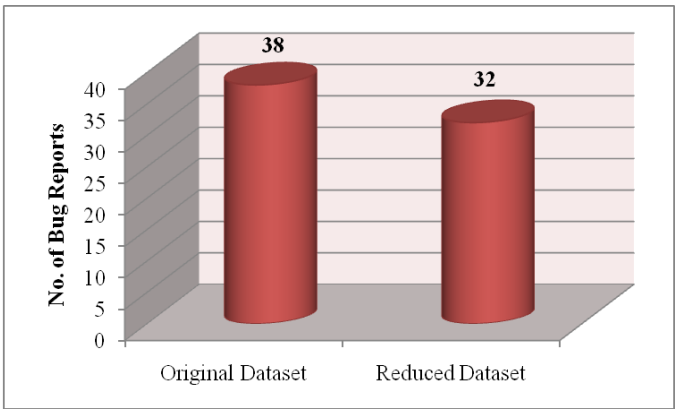

Figure 3: Training Bug Data Set

The results of data reduction for bug triage can be measured in two aspects, namely the scales of data sets and the quality of bug triage. Based on Algorithm, the scales of data sets are configured as input parameters. The quality of bug triage can be measured with the accuracy of bug triage. The first 80 percent of bug reports are used as a training set and the left 20 percent of bug reports are as a test set. The data reduction on a data set is used to denote the data reduction on the training set of this data set since we cannot change the test set.

Result Discussion

1) First, it will show how many bugs are not assigned to any developer.
2) Second, it will give complete status about the bugs to the admin so that he will come to know which bugs are assigned.
3) Third, 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 handled completely.
4) Fourth, 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.

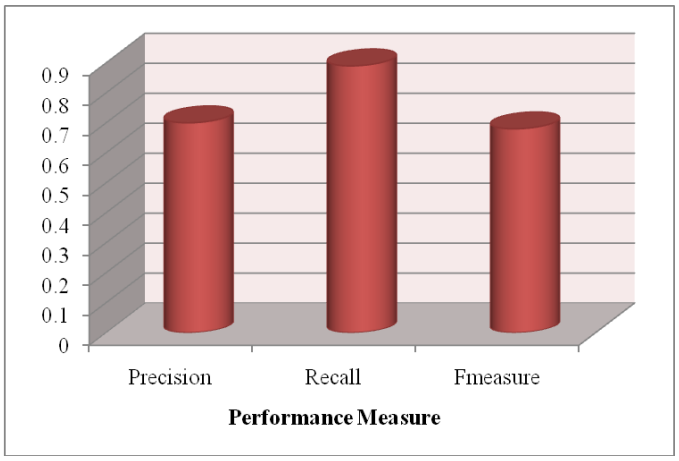

Figure 4: Performance Measure

3. CONCLUSION

Bug triage is an expensive step of software maintenance in both labor cost and time cost. The 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. The empirical investigation on the data reduction for bug triage in bug repositories was done on two large open source projects, namely Eclipse and Mozilla. This work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

FEATURE ENHANCEMENT

In future enhancement of the proposed system is to improve 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, plan to pay efforts to find out the potential relationship between the attributes of bug data sets and the reduction orders.

REFERENCES

[1] 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.
[2] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in Proc. ACM Conf. Comput. Supported Cooperative Work, Feb. 2010, pp. 301–310.
[3] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," in Proc. 7th IEEE Working Conf. Mining Softw. Repositories, May 2010, pp. –10.
[4] W. Zou, Y. Hu, J. Xuan, and H. Jiang, "Towards training set reduction for bug triage," in Proc. 35th Annu. IEEE Int. Comput. Soft. Appl. Conf., Jul. 2011, pp. 576–581.
[5] 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.
[6] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in Proc. 26th IEEE/ACM Int.Conf. Automated Softw. Eng., 2011, pp. 253–262.
[7] Kalyanasundaram Somasundaram and Gail C. Murphy, "Automatic Categorization of Bug Reports Using Latent Dirichlet Allocation," in Proc. ISEC., Feb. 2012, pp. 125–130.
[8] T. Zimmermann, N. Nagappan, P. J. Guo, and B. Murphy, "Characterizing and predicting which bugs get reopened," in Proc. 34th Int. Conf. Softw. Eng., Jun. 2012, pp. 1074–1083.
[9] S. Shivaji, E. J. Whitehead, Jr., R. Akella, and S. Kim, "Reducing features to improve code change based bug prediction," IEEE Trans. Soft. Eng., vol. 39, no. 4, pp. 552–569, Apr. 2013.
[10] H. Zhang, L. Gong, and S. Versteeg, "Predicting bug-fixing time: An empirical study of commercial software projects," in Proc. 35thInt. Conf. Softw. Eng., May 2013, pp.1042–1051.
[11] Tihana galinac grbac and goran mausa, "Stability of Software Defect Prediction in Relation to Levels of Data Imbalance,"Proceedings of the 2ndWorkshop of Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA), Novi Sad, Serbia, September 2013.
[12] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.