An Automated Approach for Software Bug Classification

Neelofar
Department of Computer Engineering
National University of Science and Technology
Rawalpindi, Pakistan
Neelofar.eme@gmail.com

Prof Dr. Muhammad Younus Javed
Department of Computer Engineering
National University of Science and Technology
Rawalpindi, Pakistan
myjaved@ceme.nust.edu.pk

Hufsa Mohsin
Department of Computer Engineering
National University of Science and Technology
Rawalpindi, Pakistan
hufsamohsin@gmail.com

Abstract— **Open source projects for example Eclipse and Firefox have open source bug repositories. User reports bugs to these repositories. Users of these repositories are usually non-technical and cannot assign correct class to these bugs. Triaging of bugs, to developer, to fix them is a tedious and time consuming task. Developers are usually expert in particular areas. For example, few developers are expert in GUI and others are in java functionality. Assigning a particular bug to relevant developer could save time and would help to maintain the interest level of developers by assigning bugs according to their interest. However, assigning right bug to right developer is quite difficult for tri-ager without knowing the actual class, the bug belongs to. In this research, we have classified the bugs in different labels on the basis of summary of the bug. Multinomial Naïve Bayes text classifier is used for classification purpose. For feature selection, Chi-Square and TFIDF algorithms were used. Using Naïve Bayes and Chi- square, we get average of 83 % accuracy.**

Key-Words: *Text mining, classification, software repositories, open source software projects, triaging, feature extraction*

1. Introduction

Data mining is the process of extracting useful information through data analysis. It is also known as knowledge discovery. Useful knowledge obtained as a result of data mining can be use to cut costs, increase revenues or both. Target data for mining purpose is categorical and numerical having data types like integer, decimal, float, char, varchar2 etc.

Data mining techniques cannot be applied to data that is not numerical or categorical. 85% of enterprise data falls in the category of non numerical or non categorical [1]. For the success of business, knowledge extraction from this unstructured data can be critical. Unstructured data is processed using text mining techniques so that it can be processed by data mining algorithms and techniques. Techniques from information extraction, information retrieval and natural language processing are used by text mining .

Classification is a function of data mining to assign classes/categories to items in a collection. Basic goal of classification is the accurate prediction of target class for each case in data. For example, loan applications can be classified into high, medium or low risks on the basis of classification model.

1.1 Mining Software Repositories

To understand constantly evolving software systems is a very daunting task. Software systems have history of how they come to be and this history is maintained in software repositories. Software repositories are the artifacts that document the evolution of software systems. Software repositories often contain data from years of development of a software project [2].

Examples of software repositories are:

a) Runtime Repositories: Example of runtime repositories is deployment logs that contain useful information about application usage on deployment sites and its execution.
b) Historical Repositories: Examples of historical repositories are bug repositories, source code repositories and archived communication logs.
c) Code Repositories: Examples of code repositories are Google code and codeforge.net that store source code of various open source projects [3].

MSR is the process of software repositories analysis to discover meaningful and interesting information

978-0-7695-4687-2/12 $26.00 © 2012 IEEE
DOI 10.1109/CISIS.2012.132

hidden in these repositories. There is a huge Software Engineering data over the course of time. MSR picks this data, processes and analyzes it, and detects patterns in this data. MSR is an open field, both in what can be mined and what one can learn from the practice. Any software repository can be mined not necessarily the code, bug or archived communication repositories.

In this paper we present an automated bug classification system. Proposed system uses Naïve Bayes text classifier to classify bugs from open bug repository. Data from eclipse and Firefox is used and maximum of 85% of precision accuracy is obtained.

2. Problem Formulation

Triaging of bugs to developer to fix them is a tedious and time consuming task. Developers are usually expert in some particular area. For example few developers are expert in GUI and others are in pure java functionality. Assigning a particular bug to relevant developer could save time as well as would help to maintain the interest level of developers by assigning those bugs according to their interest. However assigning right bug to right developer is quite difficult for tri-ager without knowing the actual class a bug belongs to. This research proposes a technique for classification of open source software bugs using the summary provided by bug reporters.

2.1 literature survey

Some of the already implemented techniques for software bugs classification are:

a) Micheal W. Godfrey, Olga Baysal and Robin Cohen presented a framework for automatic assignment of bugs to developers for fixation using vector space model [4].
b) Hemant Josh, Chuanlei Zhouang, Oskum Bayrak presented a methodology to predict future bugs using history data [5].
c) Lei Xu, Lian Yu, Jingtao Zhao, Changzhu Kong, and HuiHui Zhang proposed a technique using data mining that automatically classifies the bugs of web-based applications by predicting their bug type. They further proposed debug strategy association rules which find the relationship between bug types and bug fixing solutions [6].
d) Nicholas Jalbert and Westley Weimer proposed a system that automatically indicates whether an arriving bug report is original or duplicate of an already existing report. It saves developer's time. To predict bug duplication, system uses textual semantics, graph clustering and surface features [7].
e) Tilmann Bruckhaus provided a technique for Escalation Prediction to avoid escalations by predicting the defects that have high escalation risk and then by resolving them proactively [8].

3. Problem Solution

This section describes the proposed system for bug classification, data used for classification task and results obtained in different experiments.

3.1 Input Data

Eclipse and Mozilla firefox data is obtained from bugzilla -an open bug repository [9] [10]. Dataset of almost 29,000 record set is obtained. This data is divided into training and testing groups and experiments are performed on different set of data from these groups.

3.2 Model for prediction

When the bug is first reported to repository, it is submitted to our proposed system as shown in Fig. 1. System extracts all the terms in these reports using bag of words approach. The vocabulary is that of extremely high dimensionality and thus numbers of features are reduced by using chi-square algorithm. These features are used for training of classification algorithm which is then used for classification of bug reports. The classification algorithm used in proposed system is multinomial Naïve Bayes.

3.2.1 Pre-processing

Data pre-processing is the most important step of data mining. Data obtained from bug repositories is in raw form and cannot be directly used for training the classification algorithm. The data is first pre-processed to make it useful for training purpose. Data pre-processing is the major time consuming step of data mining and most important as well. Stop-words dictionary and regular expression rules are used to filter useless words and filter the punctuations respectively. Porter stemming algorithm is used to stem the vocabulary

3.2.2 Feature Selection

The vocabulary obtained after applying "bag of words" approach on data has very large dimensionality. Most of these dimensions are not related to text categorization and thus result in reducing the performance of the classifier. To decrease the dimensionality, the process of feature selection is used which takes the best k terms out of the whole vocabulary which contribute to accuracy and efficiency.

There are a number of feature selection techniques such as Chi-Square Testing, Information Gain (IG), Term

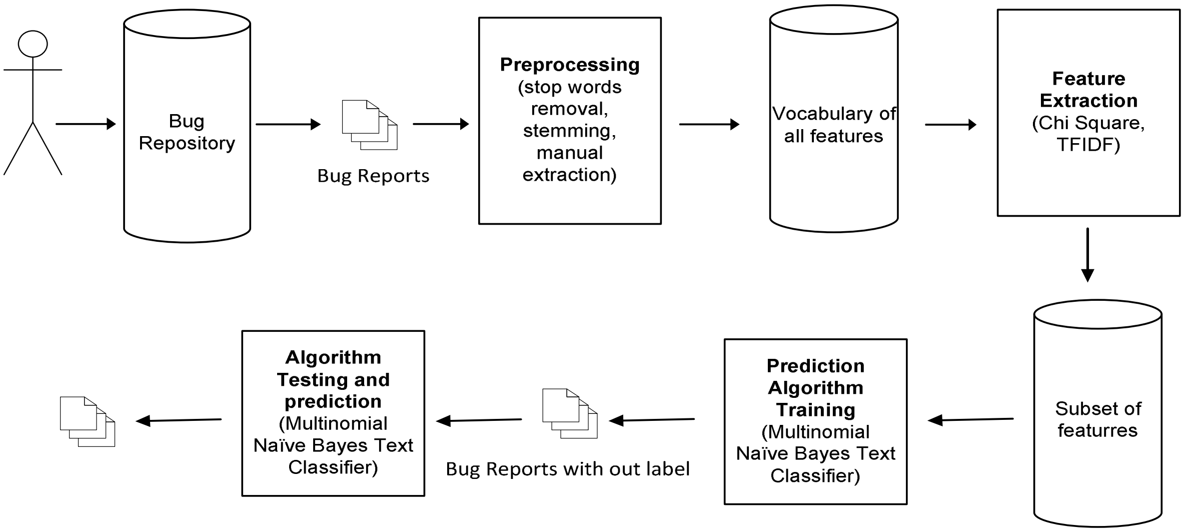
Bug Repository
Bug Reports
Preprocessing (stop words removal, stemming, manual extraction)
Vocabulary of all features
Feature Extraction (Chi Square, TFIDF)
Subset of featurres
Prediction Algorithm Training (Multinomial Naïve Bayes Text Classifier)
Bug Reports with out label
Algorithm Testing and prediction (Multinomial Naïve Bayes Text Classifier)

Figure 1. Bug Classification System

Frequency Inverse Document Frequency (TFIDF), and Document Frequency (DF). In this research, chi-square and TFIDF algorithms are used for feature selection.

3.2.3 Classifier Modeling

Text classification is an automated process of finding some metadata about a document. Text classification is used in various areas like document indexing by suggesting its categories in a content management system, spam filtering, automatically sorting help desk requests etc.

Naïve Bayes text classifier is used in this research for bug classification. Naïve Bayes classifier is based on Bayes' theorem with independent assumption and is a probabilistic classifier. INDEPENDENCE means the classifier assumes that any feature of a class is unrelated to the presence or absence of any other feature.

4. Experimental Results

Results are obtained on the basis of prediction accuracy. Prediction Accuracy is defined as
"*Ratio of the bug reports with correct class to the total number of bug reports* [6]."

For feature extraction TFIDF and Chi Square algorithms are used. Experimental results showed that Chi Square gives better results in case of bug classification from open source bug repositories. Comparison of results with TFIDF and Chi Square as feature selection algorithm is given in fig. 2. Chi Square gives higher accuracy as compared to TFIDF with same testing to training ratio.

Fig. 3 shows the effect of changing the training to testing ratio on prediction accuracy. It clearly shows that prediction accuracy increases as training to testing ratio increases. Highest accuracy is obtained when this ratio is 1:11. Increasing the training to testing ratio although increases the prediction accuracy but execution time of algorithm increases as well. So, increasing the training vocabulary data beyond a certain limit is not feasible in real time applications.

Proposed system using Naïve Bayes classifier is a probability based approach that works on the prior probability of classes and conditional probability of features in the classes. Another important model for text classification is support vector machine. Changzhu Kong, Lian Yu, Lei Xu, HuiHui Zhang and Jingtao Zhao used SVM for bug classification. Proposed technique using naïve Bayes text classifier has following advantages over the proposed system:

a) When training data is small, proposed system performs better than SVM based system of Lei Xu [6]. Training curve for SVM is much greater than Naïve Bayes and when enough training set is not given it does not perform well.
b) As far as processing time is concerned Lei system is in a disadvantage. Processing time is much higher then the other proposed technique and it grows quadratically as the number of documents increases in training set. Comparison of the two systems is given in fig. 4 and 5.

5. Conclusion

In open source bug repositories, bugs are reported by users. Triaging of these bugs is a tedious and time consuming

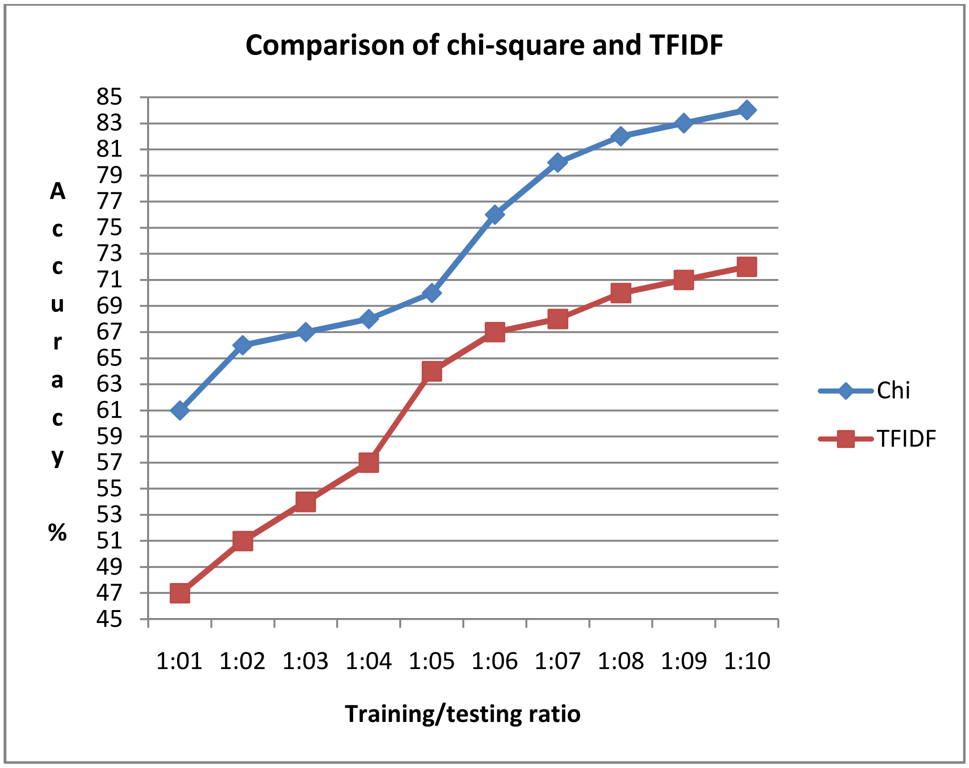

Figure 2. Comparison of Chi-Square and TFIDF

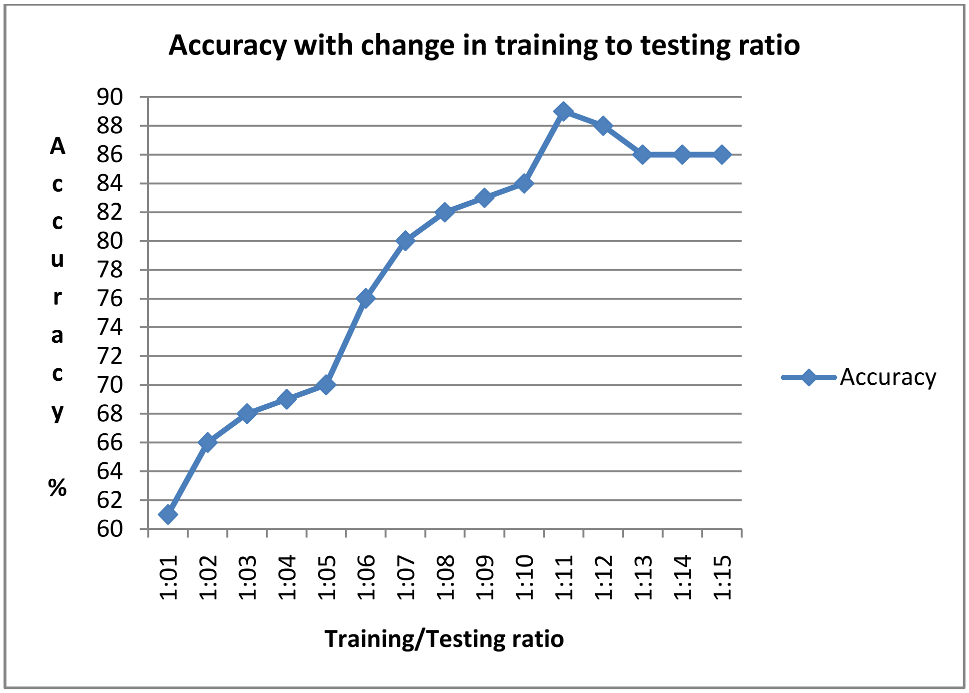

Figure 3. Accuracy with change in Training to testing ratio

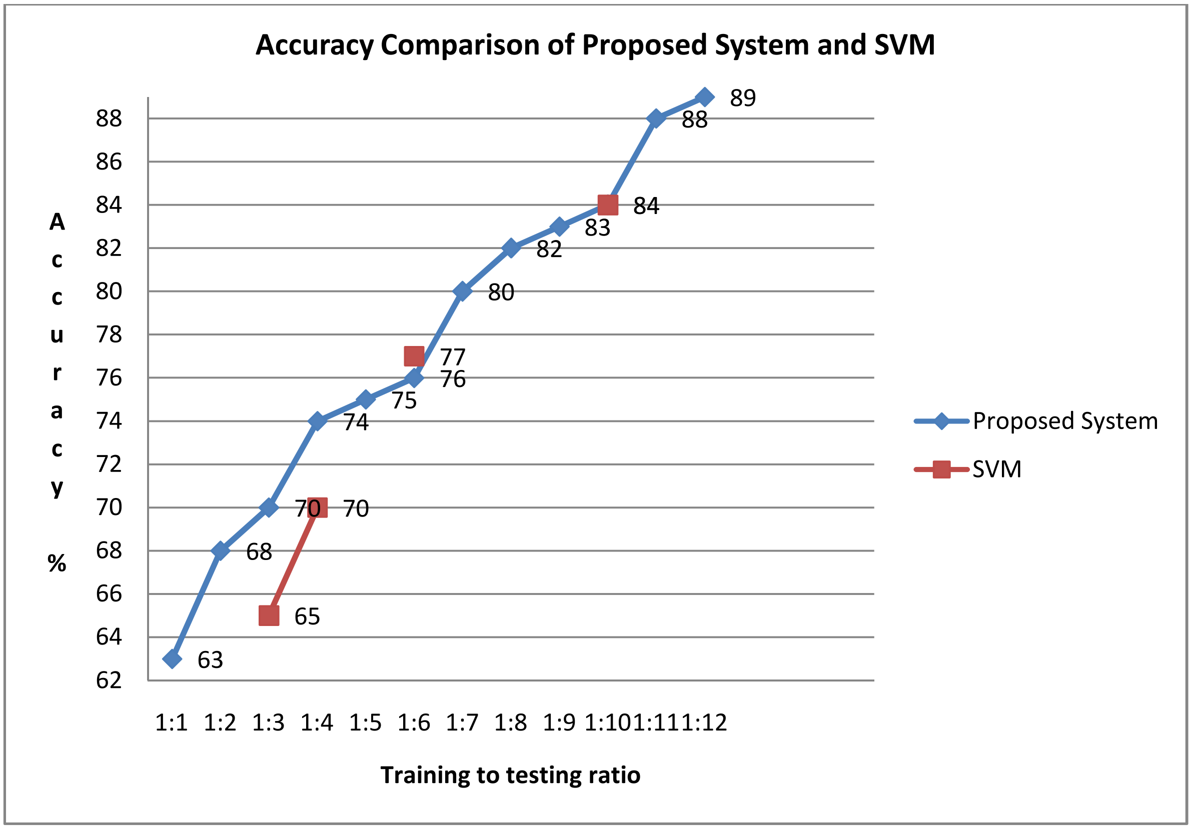

Figure 4. Accuracy Comparison of proposed system and one proposed by Lian Yu

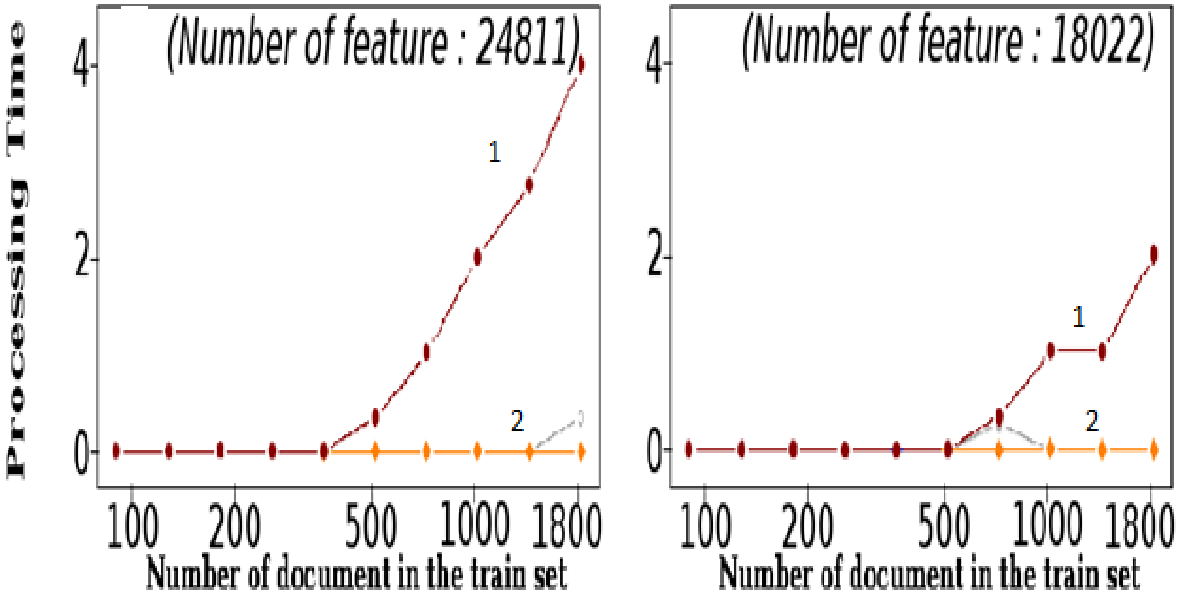

Figure 5. Comparison of Naïve Bayes (2) and SVM (1) on the basis of processing time

task. If some proper class is assigned to these bugs it would be easier to assign these bugs to relevant developers to fix them. However, as reporters of these bugs are mostly non-technical it would not be possible for them to assign correct class to these bugs. In this research an automated system for classifying software bugs is devised, using multinomial Naïve Bayes text classifier. Chi Square and TFIDF are used for feature selection. Maximum of 86% prediction accuracy is obtained.

6. Future Work

The system can be further improved by applying feature selection techniques other than Chi-Square and TFIDF. Synonym dictionary can be used so that system can tackle the issue of understanding the synonyms of the same words. For instance, if a user reports a bug related to Firefox, system should consider it Firefox whether user uses the word browser or Mozilla Firefox for it.

REFERENCES

[1] A. Hotho, A. Nürnberger and G. Paaß, "A Brief Survey of Text Mining," vol. 20, GLDV Journal for Computational Linguistics and Language Technology, 2005, pp. 19-62.

[2] A. E. Hassan, "The Road Ahead for Mining Software Repositories," *IEEE Computer society,* pp. 48-57, 2008.

[3] S. Diehl, H. C. Gall and A. E. Hassan, "Special issue on mining software repositories," in *Empirical Software Engineering An International Journal © Springer Science+Business Media*, 2009.

[4] O. B. Michael and G. C. Robin, "A Bug You Like: A Framework for Automated Assignment of Bugs.," *IEEE 17th international conference,* 2009.

[5] C. Zhang, H. Joshi, S. Ramaswamy and C. Bayrak, "A Dynamic Approach to Software Bug Estimation," in *SpringerLink*, 2008.

[6] L. Yu, C. Kong, L. Xu, J. Zhao and H. Zhang, "Mining Bug Classifier and Debug Strategy Association Rules for Web-Based Applications," in *08 Proceedings of the 4th international conference on Advanced Data Mining and Applications* , 2008.

[7] N. Jalbert and W. Weimer, "Automated Duplicate Detection for Bug Tracking Systems," in *IEEE computer society*, 2008.

[8] T. Bruckhaus, C. X. Ling, N. H. Madhavji and S. Sheng, "Software Escalation Prediction with Data Mining," in *Data Mining, Fifth IEEE International Conference*, 2006.

[9] [Online]. Available: https://bugzilla.mozilla.org/.

[10] [Online]. Available: https://bugs.eclipse.org/bugs/.