Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 5, Issue. 11, November 2016, pg.143 – 148

A Technique of Combine Feature Selection and Instance Selection for Effective Bug Triage

Aparna S. Murtadak[1], Prof. S. R. Durugkar[2]

[1]Department of Computer Engg, S.N.D.C.O.E.R.C. Yeola, Dist-Nasik,
Savitribai Phule Pune University, Maharashtra, India
[2]Department of Computer Engg, S.N.D.C.O.E.R.C. Yeola, Dist-Nasik,
Savitribai Phule Pune University, Maharashtra, India
[1] *murtadakaparna@gmail.com*, [2] *santoshdurugkar@gmail.com*

Abstract — The procedure of fixing bug is called as bug triage that expects to appropriately assign a developer to a new bug. Programming system firms pay huge cost in dealing with these software bugs. To diminish the time cost and labor cost of bug triaging, here presents an automatic approach to predict developer with significant skill to determine the new coming bug report. In proposed approach for Data reduction two data preprocessing techniques are used. It reduces the dimensions of Bug data set still maintaining the integrity of original bug data set. Diminishing the dimensionality of the data has been a difficult task in data mining but it is very important for accurate bug triage. And for Bug triage Naive Bayes and KNN algorithms are used in proposed approach. The experimental results demonstrate that the intended theme will efficiently improve the detection performance compared with earlier methods.

Keywords — Bug, Bug data reduction, Bug triage, Feature selection, Instance selection, Prediction for reduction order

I. INTRODUCTION

The procedure of extricating valuable data through data analysis is called data mining. Helpful information is obtained as a result of data mining which can be utilized to cut the expenses and time. In current programming development, software repositories are large-scale databases for storing the output of programming improvement. First stride in bug repository is to manage programming bugs but Bug fixing is an imperative and tedious procedure in programming support. Open source development projects normally consolidate an open bug repository to which both programming developers and clients can report bugs. The benefit of an open bug repository is that it might permit more bugs to be recognized and solved, enhancing the quality of the product item.

Bug triage is very important stride for bug fixing, is to allot a developer to new bug. For open source programming developments, huge numbers of bugs are created day by day which makes the triaging procedure extremely troublesome and difficult. Fundamental goal of bug triage is to allot a developer for bug fixing. Once developer is allocating to bug he will settle the bug or attempt to correct it. One of the essential reasons why bug triaging is such a long procedure is the trouble in choice of the most skilful developer for the bug kind. The bug triager, the individual who appoints the bug to a developer, must know about the exercises of the considerable number of developers in the task. It is truly essential on part of bug triager to appoint the bug report to a developer who could effectively settle the bug without need of any tossing. Therefore, the job of bug triager is

truly vital. The inspiration of this work is to decrease the extensive size of the data set and to remove the noisy and repetitive bug reports for effective bug triage. Information lessening is the procedure of diminishing the bug information by utilizing two strategies specifically, Feature selection and Instance selection which expects to get low scale and in addition quality data.

II. LITERATURE SURVEY

As new programming frameworks are getting bigger and more complex each day, programming bugs are becoming unavoidable. Bugs are the programming errors that cause noteworthy performance degradation and Bug triage is very important stride for bug fixing, is to allot a developer to new bug to fix it [1]. A developer, who is assigned to bug report, begins to fix the bug based on the awareness of past bug fixing. Normally, the developer pays efforts to comprehend the new bug report and to analyze historically fixed bugs as a kind of perspective [13]. Bug triaging helps in choosing what to do with the reported bugs. Because of the tremendously large number of bugs being accounted in Bug Repository, their classification during triaging is a boring and tedious procedure. Analysts have been for quite a while struggling to automate the bug triaging job. Decent success has also been accomplished.

Jeong et al. discover that more than 37 percent of bugs have been reassigned in manual bug triage and propose a graph based model Markov chains, which catches bug tossing history. This model has a few satisfying qualities. To begin with, it shows developer networks which can be utilized to discover group structures and to discover reasonable specialists for new task. Second, it assists to allocate master developers to bug reports [7]. To enhance the quality of bug reports, Breu et al. have manually examines 600 bug reports in open source developments to look for disregarded data in bug information [11].

Matter et al. propose an expertise model in view of source code contributions and apply in it a recommendation framework that allocates developers to bug reports. They compare vocabulary found in the developers contributions with the vocabulary found in the explanation of a bug report. And after that suggest developers whose contribution vocabulary is lexically like the vocabulary of the bug report [14]. Expansive open source developments are troubled by the rate at which new bug reports show up in the bug repository. In this paper authors present a semi-automated approach is used to the task a bug reports to a developer. This methodology applies a machine learning algorithm to the open bug archive to take in the sorts of reports every developer resolves. At the point when new bug report arrives, the classifier delivered by the machine learning procedure suggests a little number of developers appropriate to solve the bug report [2].

Park et al. change over bug triage into an improvement issue and propose a collaborative filtering approach to deal with manage diminishing the bug fixing time, which improves the recommendation quality [16]. From the above literature survey, it is concluded that early recognition and classification of bugs is important for maintaining the quality of software.

III. PROPOSED SYSTEM

The procedure of fixing bug is a bug triage that expects to appropriately assign a developer to a new bug. But manual bug triage procedure is exceptionally tedious and excessive. So to escape time cost automatic bug triage in which text classification methods are used. The issue which is addressed for this is the huge bug dataset. Also, the huge bug dataset affects the quality of the bug datasets. So to lessen the bug dataset we used Feature selection and Instance selection procedures as shown in architecture. This procedures decreases the bug data in both word and bug dimensions. The proposed system gives the high quality of data and reduces the scale of data.

Figure 1 shows the structural design of the proposed system. In the proposed system we collect the data set from open source project eclipse. This system takes Bug data set as an input. Each bug data set contains number of bug reports and every bug report contains different fields like ID, Description, Status and Operating System etc. Bug reports are unstructured data which may contain irrelevant and redundant words. Therefore, we apply the conventional text processing approach to convert the text data into a meaningful representation. After data reduction agglomerative clustering is used to make different clusters before classification, and for classification Naive Bayes algorithm is used. Output of proposed system is list of top K recommended developers for bug.

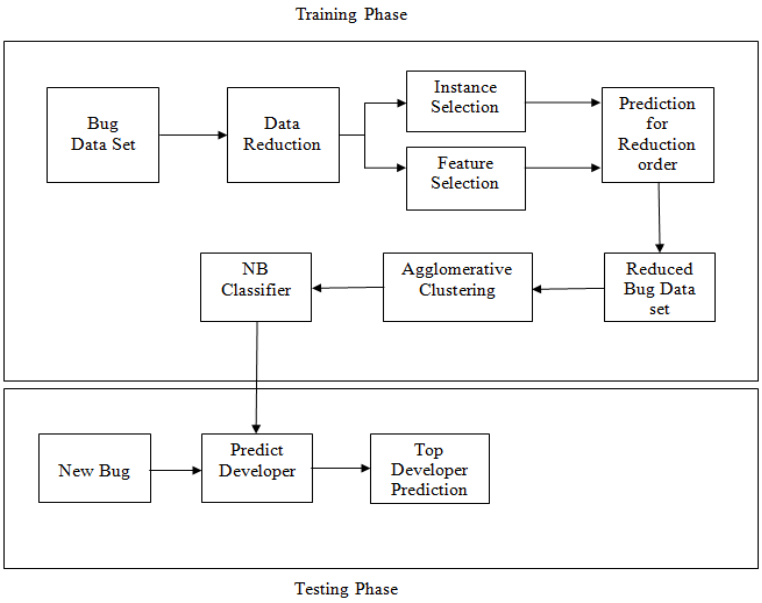

Fig 1: System Architecture

A. Pre-processing

The most vital stride of Data mining is data pre-processing. Raw data is acquired from bug repositories which can't be straightforwardly utilized for the further algorithms. Hence the information should be pre-processed to make it functional. Data pre-preparing is a monotonous stride of Data mining and most vital also. Therefore Data pre-processing is very important step in Implementation.

B. Instance Selection (IS):

Instance Selection is for acquiring a subset of relevant instances (i.e. important bug reports in bug data set). In Proposed system the purpose of Instance Selection algorithm is to reduce the size of a bug dataset by removing duplicate bug reports and keeping the originality of the bug dataset.

C. Feature Selection (FS):

This aims to acquire a subset of relevant features (i.e. important words in bug data set). In Proposed system the purpose of feature selection algorithm is to eliminate the irrelevant and redundant words from the selected bug data set, thus optimizing the efficiency of the classification and clustering algorithms.

IV. IMPLEMENTATION DETAILS

A. Dataset

To perform the experiments, we should set up the dataset for bug data reduction. In this paper bug data set of two large open source projects eclipse and Mozilla are used for experiment. Eclipse is a multi-language software development environment, including an Integrated Development Environment (IDE). And Mozilla is an open source web browser.

B. Algorithms

Following algorithms are used for data reduction in bug fixing.

- Instance Selection Algorithm:

  Input: Bug Data set

  Output: The set of selected bug reports

Processing steps:

Step 1: Read All Bug Reports.

Step 2: Calculate total unique words (U) in all bug reports

Step 3: Calculate TF value for each word for each document

Step 4: Create feature vector of each bug report as fd1, fd2..... fdn.

Step 5: Apply Pearson correlation score calculation

```
    for(i=fd1   to    fdn)
{
            for(j=i+1 to fdn)
{
            Calculate distance using Pearson correlation score calculation
}
```

Step 5.1: Apply KNN to find top similar document to current document as set (ks)

Step 5.2: Remove all ks set from corpus

}

Step 6: Repeat step 5 for all Bug reports.

- Feature Selection Algorithm:

Input: Bug Data set

Output: The set of selected words

Processing steps:

Step 1: Read all bug reports from input bug data set.

Step 2: Calculate unique words array U.

Step 3: Calculate term frequency of each unique word for every bug report.

Step 4: Re-rank all words in U.

Step 5: Calculate zero count for each word in all reports.

Step 6: Remove all words having information gain less than T.

Where T is user define threshold value.

Step 7: Calculate performance of Feature selection algorithm

Step 8: Display updated new bug data.

V. RESULT ANALYSIS

In this system, two data pre-processing techniques are used for data reduction. To perform the experiments, I have set up the dataset for bug data reduction from open source project Eclipse and Mozilla. Below figure 2 shows result of Bug Triage accuracy with Instance selection and Feature selection on Eclipse in term of Precision, recall and F1 in comparison with Base paper result.

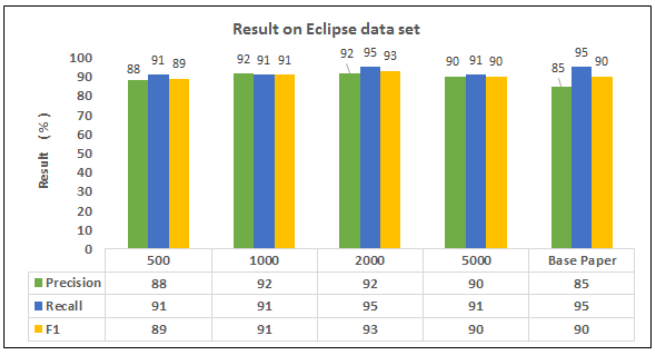

Fig 2: Result on Eclipse data set

Below figure 3 shows result of Bug Triage accuracy with Instance selection and Feature selection on Mozilla in term of Precision, recall and F1 in comparison with Base paper result.

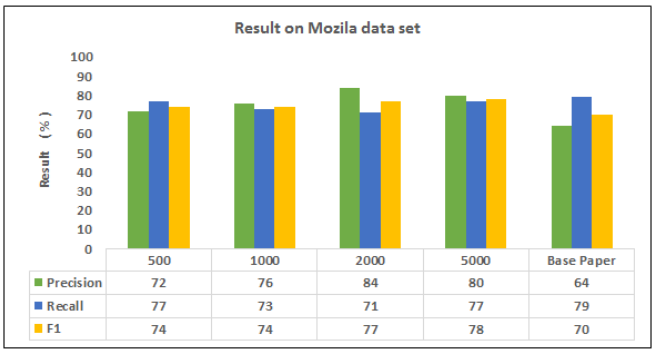

Fig 3: Result on Mozilla data set

VI. Conclusion

In this paper i have focused on data reduction for effective Bug triage. Bug triage is a costly stride of programming upkeep in both work cost and time cost. In this paper the proposed system focused on lessening bug information set with a specific goal to have less size of data and quality data, for that two data reduction techniques are used that is Instance Selection and Feature Selection. The Proposed system can be used any large open source development for large scale bug data. This reduces the large scale of bug data and provides high quality data. This data reduction is necessary for effective bug triage. And for bug triage KNN and Naive Bayes algorithms are used in this paper. In future we can use this system for any large project for data reduction and effective bug triage.

Acknowledgement

I am very much thankful to all authors those are mentioned in the references. I would like to thanks my guide for their constant support and motivation. Also I would like to thanks PG coordinator and HOD of computer department who helped me in presenting this paper.

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] 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.

[3] D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.

[4] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 249–283, 2013.
[5] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," J. Mach. Learn. Res., vol. 3, pp. 1157–1182, 2003.
[6] Q. Hong, S. Kim, S. C. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in Proc. 27th IEEE Int. Conf. Softw. Maintenance, Sep. 2011, pp. 323–332.
[7] 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.
[8] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in Proc. 34th Int. Conf. Softw. Eng., 2012, pp. 25–35.
[9] 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.
[10] R. J. Sandusky, L. Gasser, and G. Ripoche, "Bug report networks: Varieties, strategies, and impacts in an F/OSS development community," in Proc. 1st Intl. Workshop Mining Softw. Repositories, May 2004, pp. 80–84.
[11] 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.
[12] S. Kim, K. Pan, E. J. Whitehead, Jr., "Memories of bug fixes," in Proc. ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2006, pp. 35–45.
[13] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 618–643, Oct. 2010.
[14] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reports using a vocabulary-based expertise model of developers," in Proc. 6th Int. Working Conf. Mining Softw. Repositories, May 2009, pp. 131–140.
[15] S. Just, R. Premraj, and T. Zimmermann, "Towards the next generation of bug tracking systems," in VL/HCC, pages 82–85, 2008.
[16] J. W. Park, M. W. Lee, J. Kim, S. W. Hwang, and S. Kim, "Costriage: A cost-aware triage algorithm for bug reporting systems," in Proc. 25th Conf. Artif. Intell., Aug. 2011, pp. 139–144.

