Effective Bug Triage with Data Reduction

S. R. Birajdar
Dept.of Computer Science and Engineering
VVP Institute of Engineering and Technology,
Soregaon, Solapur, Maharashtra, India

H. B. Torvi
Assistant Professor
Dept.of Computer Science and Engineering
VVP Institute of Engineering and Technology,
Soregaon, Solapur, Maharashtra, India

ABSTRACT
As human beings we all make mistakes and these mistakes are reflected as defects in the software product. These defects make the software fail which are due to our limitations as human beings. When the testing is done, the reasons for failure are identified and the defects are found. Then the defects are corrected. This is an iterative process- you need to test the software, fine defect, correct the code, and test software again. The defect has to be removed by developer. To remove defect which is not easy, Most of the organization spend 40% of cost to remove defect. The process of fixing or remove bug is bug triage. Remember every mistake in manual bug triggering, even those rated with the least priority. In order to reduce time, manual boom trials are applied to price, text classification techniques to take automated bug triage.

In this paper, deal with the problem of data reduction for bug triage, i.e., how to reduce the scale and improve the quality of bug data. For the same combine instance selection and feature selection to reduce data scale on the bug dimension and the word dimension.

Keywords
Bug Triage, Bug Report, Instances Selection, Feature Selection, Data Reduction

1. INTRODUCTION
Fuzzy needs, software complications, programming errors, communication gap, documentation errors, standard deviation, lack of design, lack of designing experience, unreal time schedules for development, bugs in software due to last minute changes in error. Fixing bug is time-consuming of software development process [1]. Late bug fixing[4] can cause software failure problems. Bug fixing is an important aspect of open source software like Eclipse and Mozilla [2]. In software projects, usually a database is maintained to collect and manage the large number of bug reports. This database is known as a bug repository or a bug tracking system [5]. Bug repositories are used for all kind of software, but they are mostly used for open source systems where the developers, users, tester and other team members are distributed all over the world. The purpose of the fault report is to reflect the bug and explain it to the developers. The bug report must comply with some guidelines, avoid duplicate bug reporting, always test the latest available build, state useful facts, do not think of votes or complain, flag security / privacy vulnerability. Bug report records summary is to make the report searchable and uniquely identifiable, Overview/Description, Steps to Reproduce, Test result, Environment setup, configuration ,any additional information. Bug Triaging involves categorizing the bug according to bug type, assign severity and priority and most important is assign the bug to effective developer who can solve bug The bug report should be clear and concise. Report an individual problem with each bug. If there is more than one problem in a single bug report, you can't close it until all issues are resolved.

The manual triage process of assigning bug to developer is error prone [6] and time consuming [16]. In case of Open Source Software Development where developers are geographically separate location and different time zones it is more difficult task to assign developer for bug fixing. But, still there was a problem with bug report are namely large scale and low quality due to miscommunication ,confusion ,lack of clarity. Report each bug as a separate issue. On one hand for open source project, an average 30 new bug are reported per day [13]. On the other hand bug reports suffer from low-quality, noise and redundancy.

This paper is having sub sections are as follows. We start with an Introduction. In Section 2 described Related work. Section 3 Proposed system and its system architecture. Algorithms in Section 4. Section 5 defines the experimental results. Finally Section 6 represents conclusions and feature works.

2. RELATED WORK
To examine the relationship between bug data, Sandusky et al. [7] form a bug report network to monitor to investigate the quality of bug data, Zimmermann et al. [8] Developer and three users Design Questionnaire Open source project. After analyzing the questionnaire, He makes good bug reports and displays them Train the classifier to identify the bug quality and correct the report. Duplicate bug reports weaken Quality of bug data by delaying handling costs to find bugs for duplicate bug reports Wang et al. [9] represents, a natural language processing method to adapt the execution information; Sun et al. [10] design a duplicate bug identification method by optimizing a recovery function on multiple features.

To determine a suitable developer of bug trials to fix a new bug, e.g. bug fixes. Cubranic and Murphy [11] first propose the problem of automatic bug triage is to reduce the cost of manual bug triage [3]. They use text classification techniques to predict related developers. Anvik et al. [12] examine several techniques on bug triage, including data preparation and typical classifiers. Anvik and Murphy [13] worked to reduce efforts to create bug-related recommendations by making development-oriented recommendations. Park et al. [15] design to convert bug trigger to optimization problem and offer a collaborative filtering approach to reduce bug fixing time [4]. In addition to studying the relationship between them Bug report, Hong et al. [14] developers Create Social Networks Examine the cooperation between developers Mozilla project's bug data or developer social network The developer is able to understand the community And the evolution of the project, map the bug priority.

3. PROPOSED SYSTEM

The proposed system, address the problem of data reduction and improve the quality to facilitate the process of bug triage.

The aim is to build effective bug report/bug data and assign bug to appropriate developer to fix bug. Bug triage is most lengthy and difficult in selecting of developer for bug fixing. The expert person who has domain knowledge will assign bug to developer. If in case assign developer could not resolve it, it is assign to another developer [6]. This would increase both time and money. Thus, it is really important part to assign the bug to developer who could successfully fix bug.

As shown in figure 1, in the proposed system, we combine instance selection and feature selection algorithms decrease the bug dimensions and word dimensions.Reduced bug data includes reporting bugs related to native bug reports and providing similar information to key word and original bug data. Historical bug sets the data base based on bugs fixed by the developer, our system can predict the most appropriate developer to solve a new bug and reduce the time taken for the fixing process.

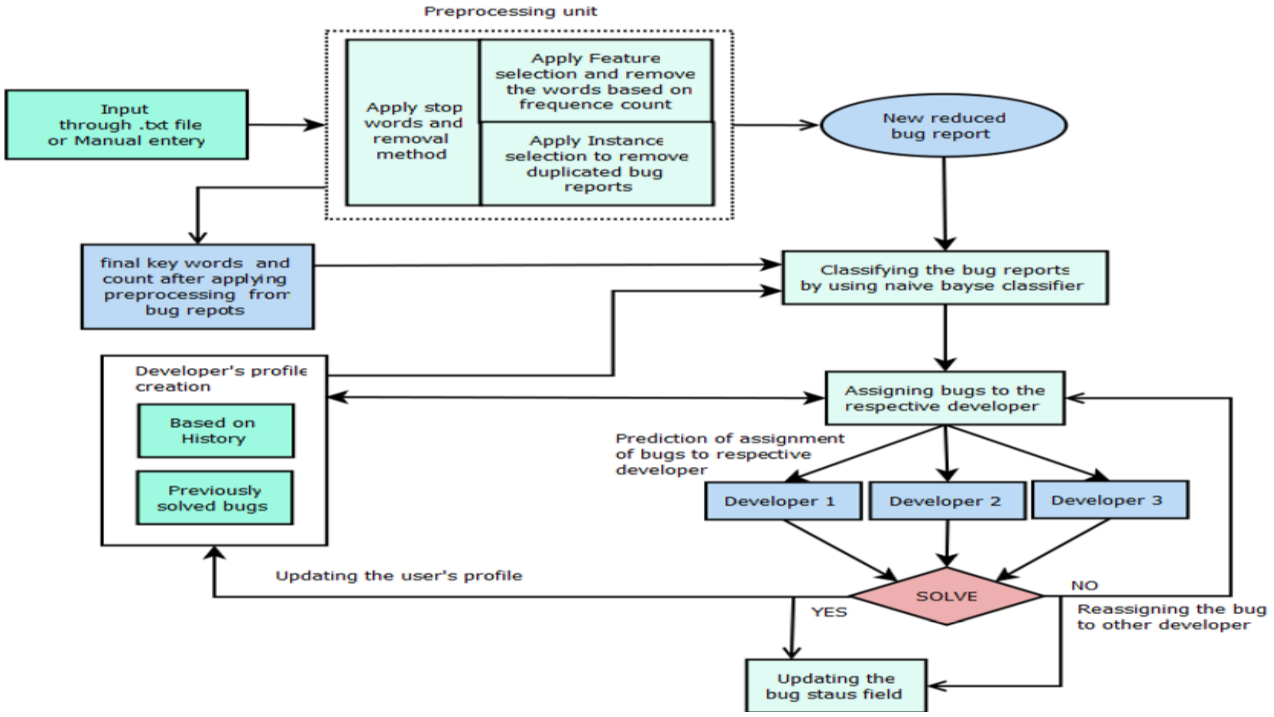

Fig 1: Bug Triage System for prediction of developers list

4. ALGORITHMS

Proposed system consists of CHI Square for feature selection and Cosine Similarity for instances selection algorithms that help in predicting developers. The algorithms used are as follow.

4.1 CHI-Square Test

The chi-square test is an important test for independence to determine the dependency of two variables. Where, chi-square test is only applicable to categorical or nominal variable. From the definition, of chi-square we can easily deduce the application of chi-square technique in feature selection [5]. Lets you have a target variable and some other features that describes each data sample. Now, we calculate chi-square relation between every feature variable and the target variable and observe the existence of a relationship between the variables and the target. If the target variable is independent of the feature variable, we can reject that feature variable. If they are dependent, the feature variable is accepted. The formula for the chi-square statistic used in the chi square test is [17].

$$X^2 = \sum \left[\frac{(O_{r,c} - E_{r,c})^2}{E_{r,c}} \right]$$

O is observed value and E is expected value. The summation symbol means that calculation for every single data item in your data set. The chi-squared statistic is a single number that tells you how much difference exists between your observed counts and the counts you would expect [5]. It includes steps in determining the association and they are as follow.

1) The degrees of freedom (DF) are equal to:

$$DF = (r-1) * (c-1)$$

Where r is the number of levels for one variable, and c is the number of levels for the other variable.

2) The expected value counts are computed separately for each level of one variable at each level of the other categorical variable. Compute r * c expected frequencies, according to the following formula [19].

$$E_{r,c} = (n_r * n_c)/n$$

where $E_{r,c}$ is the expected value for level *r* of Variable A and level *c* of Variable B, n_r is the total number of sample observations at level r of Variable A, n_c is the total number of sample observations at level *c* of Variable B, and n is the total sample size.

3) The P-value is the probability of observing a sample statistic as extreme as the test statistic. Since the test statistic is a chi-square, use the Chi-Square distribution calculator to assess the probability associated with the test statistic. Use the degrees of freedom computed above.

4) If the sample findings are unlikely, given the null hypothesis, the researcher rejects the null hypothesis. Typically, this involves comparing the P-value to the significance level, and rejecting the null hypothesis when the P-value is less than the significance level.

4.2 Cosine Similarity

Cosine similarity is a calculate similarity between two vectors. Let's say for example vector1 which is dataSet1 and vector 2 which are dataSet2. Now, going to decide how close these two vectors are to each other by calculating one function of those two vectors, namely the cosine of the angle between two vectors. Let's say for example vector1 which is dataSet1 and vector 2 which are dataSet2. Now, going to decide how close these two vectors are to each other by calculating one function of those two vectors, namely the cosine of the angle between them. We check the similarity between two date sets of report. If the similarity is greater than the threshold value, then the report is stated as duplicate and is given as a reference of previously solved report else it is passed to NB Classifier. Cosine similarity is finding using below formula.

$$\cos\theta = \frac{AB}{|A||B|}$$

4.3 NB-Classifier

NB-Classifier algorithm is used to predict developers. Along with this data the input also contains keyword count of new bug report [11]. By using these counts we calculate prior, maximum posterior probability the prediction order of developer is decide.

5. EXPERMENTAL RESULTS

The accuracy of our trained bug data set can be measured by using the formula

Accuracy $_k$= correct relevant developers/all bug reports in test data sets.

K is size of recommendation list. To improve the quality of bug triage, we follow to use recommendation list. A list with size k can provide k developer as the prediction result for each new-coming bug report. Table 1 shows bug data set for Mozilla of three different sizes. Bug data set contains bug id, bug summary, bug description and label as developer.

Table 1: Accuracy of practice Data Set on Mozilla

	Name	DS-M1	DS-M2	DS-M3
Mozilla	Label developers	20	40	60
	Correct Developers	17	30	47
	Accuracy (%)	85	65	78

The accuracy rate is the most significant evaluation criterion for bug triage since it measures the quality of prediction.

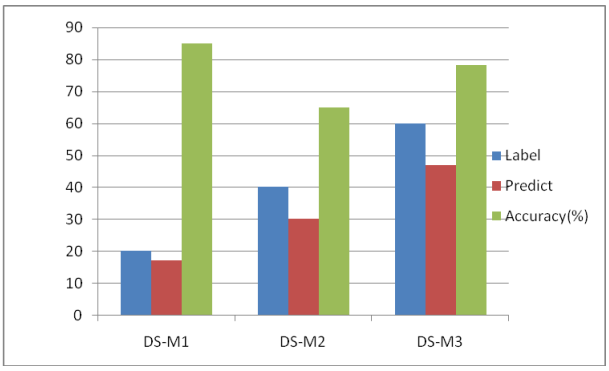

Fig 2: Accuracy for Dataset Mozilla

Precision and Recall are the basic measures used in bug resolved count and finding duplicate bug report.

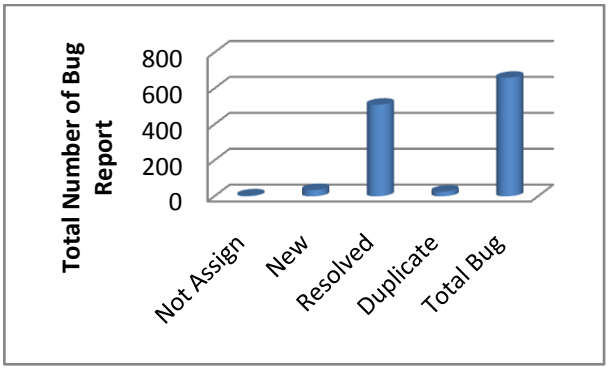

Fig 3: Experimental Bug status out of Total bug

Figure 3 shows number of not assign bug, new, resolved, duplicate bug out of total bug. Precision is the ratio of resolved bug count to total number of bug. Precision can be seen as measure of exactness or quality i.e. correctness. The Precision and recall value for the trained data set can be calculated by using the formulae.

Precision = Resolved Bug Count / Total Bug Count

Recall is the ratio of duplicate bug found to total resolved bug report. That is recall is the fraction of duplicate bug report that are successfully find. Recall is a measure of completeness or quality.

Recall = Duplicate Bug Count / Resolved Bug Count.

To balance the precision and recall value, F- measure is defined as

F-Measure = 2 * (precision * recall / precision + recall)

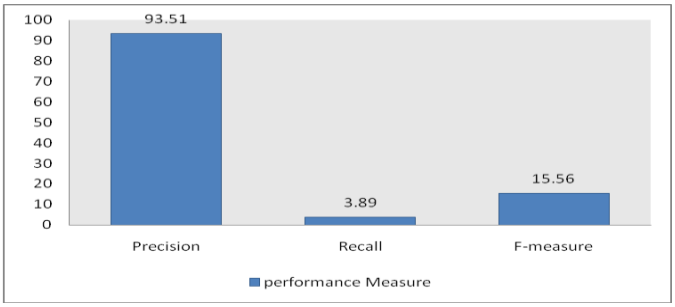

Fig 3: Comparison Graph for Precision, Recall, F-measure

6. CONCLUSION

The proposed system combines the feature selection algorithm (FS) and instance selection algorithm (IS) in order to reduce the scale of bug data sets as well as improve the data set quality. The proposed system performance is verified using Mozilla bug data set. To demonstrate the effectiveness, scales of data set is reduced by using data reduction technique in order to decrease the time and labor cost, improve the accuracy of bug triage with high-quality bug data in software development and maintenance.

The future work of the proposed system is to improve the results of data reduction in bug triage to explore how to prepare a high value bug data set and deal with a domain-specific software task. To predict reduction orders, try to identify potential links in the characteristics of bug data sets and decrease orders.

7. REFERENCES

[1] W. Zou, Y. Hu, J. Xuan, and H. Jiang, "Towards training set reduction for bug triage," in Proc. IEEE 35th Annual CS and Application Conference, Washington, DC, USA: IEEE Computer Society, 2011.

[2] Thomas Zimmermann, Rahul premraj, Jonathan Sillito, Silvia Brell, "Improving Bug Tracking Systems", ICSE'09, May 2016.

[3] J. Anvik,"Automatic bug report assignment," in Proc 28th International Conference on Software Engineering. ACM, 2006.

[4] H. Zhang, L. Gong, and S. Versteeg, "Predicting bug-fixing time: An empirical study of commercial software projects," in Proc. 35th Int. Conf. Softw. Eng, May 2013.

[5] Shanthi Priya Duraisamy, Laxmi Raja, KalaiSelvi Kandaswamy,"An Approach for Predicting Bug Triage using Data Reduction Methods"in International Journal of Computer Applications ,November 2017

[6] Gaeul Jeong, Sunghun Kim,Thomas Zimmermann," Improving Bug Triage with Bug Tossing Graphs",in Proc. Joint Meeting 12th Eur. Softw.Eng.Conf.17th ACM SIGSOFT Symp. Found. Softw. Eng., Aug.2009.

[7] 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.

[8] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schr€oter, and C. Weiss, "What makes a good bug report?" IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 618–643, Oct. 2010.

[9] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in Proc. 30th Int. Conf. Softw. Eng., May 2008, pp. 461–470.

[10] 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.

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

[12] 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.

[13] 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.

[14]Q. Hong, S. Kim, S. C. Cheung, and C. Bird, Understanding a developer social network and its evolution," in Proc. 27th IEEE Int. Conf. Software Maintenance, Sep. 2011, pp. 323–332

[15] 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.

[16] Jifeng Xuan, He Jiang , Member ,IEEE, Yan Hu, Zhilei Ren, Weiqiu Zou, Zhongxuan Luo et al: Towards Effective Bug Triage with Data Reduction Techniques, in IEEE Transactions, vol.27, No.1 January 2015

[17] S.Shibaji E.J. Whitehead, Jr., R. Akella, and S. Kim, Reduction features to improve code changes based bug prediction, IEEE Trans. Soft. Eng., vol. 39, no. 4, pp.552-569, Apr.2013.

[18] Mamdouh Alenezi and Kenneth Magel: Efficient Bug Triaging Using Text Mining in 2013 Academy Publisher