An Approach for Predicting Bug Triage using Data Reduction Methods

ShanthiPriya Duraisamy
Assistant Professor,
Dept of Computer Science & Engineering,
Karpagam Institute of Technology,
Coimbatore – 641 105,
TamilNadu, India

Laxmi Raja
Assistant Professor,
Dept of Computer Science & Engineering,
Karpagam Institute of Technology,
Coimbatore – 641 105,
TamilNadu, India

KalaiSelvi Kandaswamy
Assistant Professor,
Dept of Computer Science & Engineering,
Karpagam Institute of Technology,
Coimbatore – 641 105,
TamilNadu, India

ABSTRACT
Most of the software companies need to deal with large number of software bugs each and every day. Software bugs are inevitable and fixing software bugs is an expensive task. The proposed system employs the combination of data reduction techniques that is feature selection algorithm (FS) and instance selection algorithm (IS) in order to shrink the bug data set and also to upgrade the accuracy of bug triage. Predictive model is used to determine the order of reduction techniques for a new bug data set, i.e., to choose between FS to IS or IS to FS. The aim of effective bug triaging software is to assign potentially skilled developers to new coming bug reports. To decrease the manual and time cost, text classification techniques are applied to accomplish automatic bug triage approach aims to precisely predict the developer to solve or fix the new bug report. The proposed system performance is verified using Mozilla bug data set. To exhibit 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.

Keywords
Mining Software Repository, Data Management in Bug Repository, Bug Data Reduction, Feature Selection, Instance Selection, Bug Triage

1. INTRODUCTION
Software bug fixing process is an important and expensive task in software development and maintenance [1].In software development [5], large databases are used to store the details of bugs. This database is known as bug repository or bug tracking system. Bugzilla is such an open source bug repository [3], which is used by many large software companies for open source projects i.e., Mozilla [11]. Bugs are maintained as a bug report which records textual description to reproduce the bugs. Based on the bug tracking system, the existing bugs are easily maintained and fixed by the developers. By using data mining techniques [18], the real world software engineering problems can be solved with some useful information stored in bug repository.

Each bug reports should be assigned to relevant developer who could fix it [21]. This assignment process is known as Bug Triage. The traditional bug repositories used the human triage to fix the software bugs. Due to large number of daily bugs and lack of expertise of all the bugs, manual triage is an expensive in time cost and labor cost, low in accuracy. To overcome the limitations of existing work, an automatic bug triage approaches proposed [19]. This approach applies the text classification techniques in order to predict the relevant developer for bug reports without tossing.

Cubranic and Murphy [4] proposed supervised learning technique (NB Classifier) to assist in bug triage by using text categorization to predict the relevant developers. Wang, Zhang, Xie, and Sun [9] proposed an approach Execution information similarities (E-S) to detect the duplicate bug reports with the natural language information. They initiatively apply the Classification-based heuristic technique for labelling the bug reports. A classification model should be designed to investigate the relationship among the data's in bug data set and to check the quality [11, 16].

Anvik, L. Hiew, and G. C. Murphy [1] extend the machine learning approaches. They describe the bug triage as semi-supervised approach which updated with weighted recommendation list; based on the probabilistic view the relevant developers are employed to the human triage [5]. Matter, et al compared the text classification approaches and investigates the competence model of developers for bug triage [10].Jeong, Kim, Zimmermann introduced a tossing graph model based on Markov property from the conception of reassigns the bug reports to other developers [8].

Kim, et al [9] proposed the defect prediction model used to predict the defect- proneness (buggy or clean) of different software artifacts such as source code, file, a class or a module. Shivaji and colleagues [14] proposed the feature selection techniques to predict the software bugs. Fu.Y, Zhu.X, and Li.B [7] investigated to obtain the accurate prediction model with minimum cost by labelling most informative instances. In contrast to these papers, our paper aims to employ the information gain algorithm to improve the software quality of bug data prediction.

In this paper, we proposed the data reduction techniques and automatic bug triage approach. Here, the reduction techniques using the combination of the instance selection algorithm (IS) and feature selection algorithm (FS). Our Mozilla bug data set reduction applies instance selection (Removes unnecessary reports) [17] before or after feature selection (Removes unnecessary words) [13]. These approaches are used to reduce the data scale and also improve the accuracy of bug data set. The reduced bug data contain fewer bug data than the original bug data and provide similar information over the original bug data.

The order of applying the reduction techniques may affect the result of bug triage approach. In this paper, we propose a Predictive model in order to determine the order of bug data reduction techniques, i.e., FS to IS or IS to FS. To decrease the manual triager cost, text classification technique i.e., Naive Bayes is used to predict correct developer to solve and fix the bug reports [12]. The proposed system performance is verified using Mozilla bug data set [11].After reducing the training set, the accuracy of bug data is measured as 78%. The result shows that the experiment on reduce training sets can obtain better accuracy than that on original training set.

The remainder section of this paper is organized as follows: Section 2 presents the proposed methodology. Section 3 presents the experimental results and discussion. In Section 4 we briefly conclude this paper and present our future work.

2. PROPOSED METHODOLOGY

In this section, we present the data reduction techniques to reduce scales of bug data set. The main goal of our work is to combine the instance selection and feature selection in correct order to remove the noisy, redundant and non-informative bug reports.

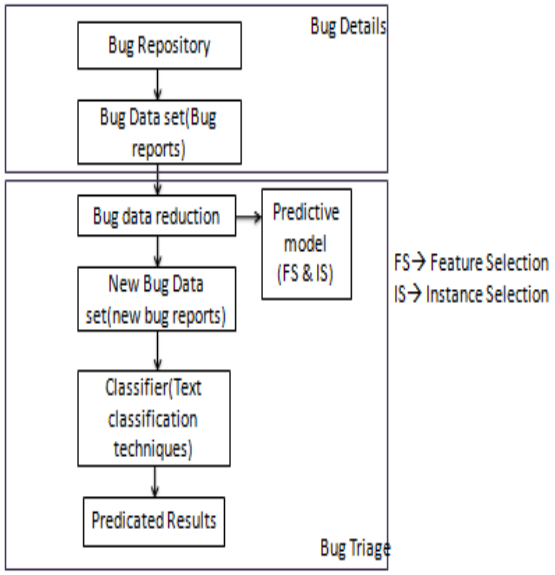

Fig.1 System Architecture

2.1 Bug Details

The bug details consist of bug repository and bug reports. In a bug repository, a bug is sustained as a bug report, which traces the textual illustration to repeat the bug and updates according to the status of bug fixing.

2.2 Bug Repository

A bug repository is a typical software repository, for storing details of bugs, e.g., a popular and open source bug repository, Bugzilla [2].Large software projects deploy bug repositories is also called as bug or issue tracking systems, which is used to support information collection and to assist developers to handle bugs. Each bug is maintained as a bug report, which traces the documentary description of reproducing the bug and revises according to the significance of bug fixing. The use of bug repository can improve the development process and quality of software produced. It presents a data platform to sustain many forms of assignment on bugs, e.g., defect prediction, bug localization and reopened bug analysis.

2.3 Bug Report

A recorded bug is called a bug report or bug data. It has multiple items for detailing the information of reproducing the bug. In a bug report, the outline and the report are two key items about the information of the bug, which are traced in natural languages. Summary denotes the general statement for identifying a bug and description gives the details to reproduce the bug [22].The bug report may also contain other items also, such as Product, Platform, and Importance.

2.4 Bug Triage

The method of allocating a correct developer for renovating the bug is called bug triage. Once the bug report is formed, a bug triager allocates the bug to a developer who can fix this bug and developer is recorded in an item assigned-to without any tossing.

2.5 Bug Data Reduction

By employ the grouping of feature selection and instance selection algorithms to get rid of unwanted and non-informative bug reports. With the experience in text categorization methods, an instance in bug triage specifies bug reports while a feature in bug triage indicates the bug words. The vital goal of our work is to reduce the text matrix with two dimensions namely, bug report dimension and word dimension.

The two-phase combination of instance selection and feature selection algorithms are employed to reduce the bug data set on two dimensions i.e., bug report dimension and word dimension. To determine the order of data reduction technique, the predictive model is applied by the proposed system. This model helps to predict the correct order i.e., FS to IS or IS to FS in order to reduce the labor cost and time cost. The reduced bug data contain fewer bug data than the original bug data and gives related information over the original bug data.

2.6 Feature Selection

Feature selection is a pre-processing method for choosing a diminished set of features for huge-scale data sets [4,17].The pre-processing techniques are tokenization, stop word removal, stemming process and vector space model. The tokenization method is used to tokenize the summary and description of the bug reports into word vectors. Non-alphabetic words and special character are removed to avoid the noisy bug words. Stop word removal technique remove the stop words in high frequency and provide no helpful information for bug triage. Stemming technique uses porter stemming algorithm for reducing inflected words their word stem/root form. Vector space model /Term vector model is an algebraic model for representing text document as vector of identifier. The minimized set is considered as the representative features of the original feature set[15].

The four well-performed algorithms are chosen in text data [13, 19] and software data, namely Information Gain (IG)**, $\chi 2$ statistic (CH)**[14], Symmetrical Uncertainty attribute evaluation (SU), and Relief-F Attribute selection (RF).

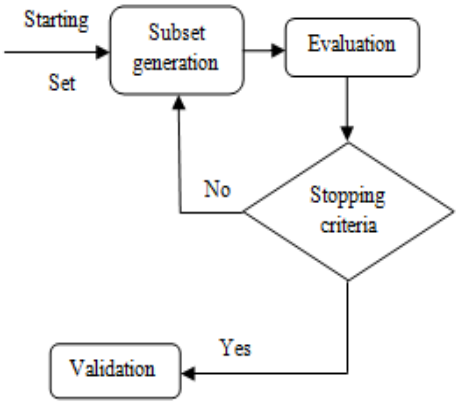

Fig. 2 General Feature Selection Structure

Based on feature selection, words in bug reports are organized according to their feature importance and a given number of words with large values are selected as representative features.

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 criterion normal random variables. It is a unique case of the gamma distribution and the most widely used probability distributions in inferential statistics. If Z_1,.....,$Z_k$are independent, standard normal random variables, then the sum of their squares,

$$Q = \sum_{i=1}^{k} Zi^2 \qquad (1)$$

is distributed according to the chi-squared distribution with *k* degrees of freedom. This is usually denoted as

$$Q \sim X^2(k) \text{ or } Q \sim X_k^2 \qquad (2)$$

where k is a positive integer that specifies the number of degrees of freedom (i.e. the number of *Zi*'s). The Chi- squared attribute evaluation evaluates the worth of a feature by computing the importance of the chi- squared gauge with respect to the class. The initial hypothesis $H_0$is the assumption that the two features are dissimilar and it is checked by chi-squared formulae:

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{c} (O_{ij} - E_{ij})^2 / E_{ij} \qquad (3)$$

where O_{ij} is the observed frequency and E_{ij} is the expected (theoretical) frequency, asserted by the null hypothesis.

2.7 Instance Selection

Instance selection is methods to diminish the number of instances by eliminate noisy and redundant instances [19]. An instance selection algorithm can give a condensed data set by eliminating non-representative instances. There are four instance selection algorithms, namely **Iterative Case Filter (ICF)**[23], Learning Vectors Quantization (LVQ), Decremental Reduction Optimization Procedure (DROP), and Patterns by Ordered Projections (POP).In the proposed the iterative case filter(ICF) algorithm defines local set L(X) which contains all cases inside largest hyper sphere centred in X such that the hyper sphere contains only cases of the same class as a instance X. The properties of ICF defined as

- Coverage of a case is the set of target problems that it can be used to solve.

$$\text{Coverage}(X) = \{X' \leq T : X \leq L(X')\} \qquad (4)$$

- Reachability of a target problem is the set of cases that can be used to afford a solution for the target.

$$\text{Reachability}(X) = \{ X' \leq T : X' \leq L(X) \} \qquad (5)$$

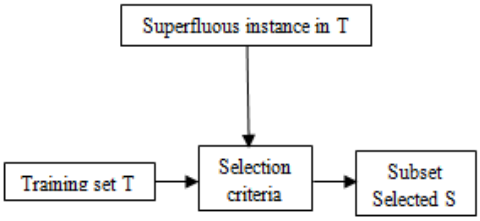

Fig.3 General Instance selection Structure

ICF algorithm eliminates each instance X for which the reachability (X) is bigger than coverage (X). For every instance in T this procedure will be repeated.

Algorithm: 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,
  1. apply FS → *n* words of *T*
  2. calculate objective values for all the words
  3. select the top n_F words of *T*
  4. generate a training set T_F
  5. apply IS →m_I bug reports of T_F
  6. terminate IS when the number of bug reports is equal to or less than m_I
  7. Generate the final training set T_{FI}.

Output:

- reduced data set T_{FI} for bug triage

2.8 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. The reduced bug data can be evaluated according to two criteria: Scale of a data set and accuracy of bug triage.

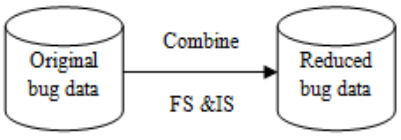

Fig.4 Data Reduction

2.9 Classifier (Text Classification Technique)

The text classification technique is used to predict the developers for bug reports [20]. A classifier can be trained only once with training data set in order to face many new bug data sets i.e., training such a classifier once can expect the reduction orders for all the new data sets without checking both the orders.

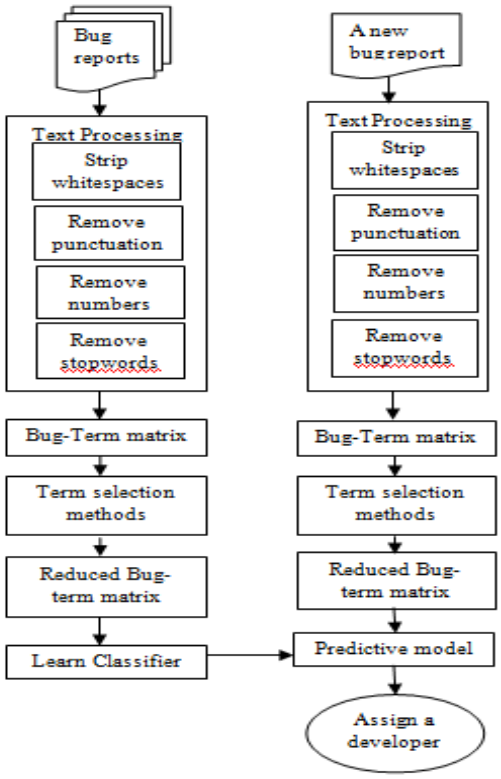

Fig.5 A Classification Approach

The text classification techniques for bug triage algorithms are Support Vector Machine (SVM), K-Nearest Neighbour (KNN) and **Naive Bayes**[19]. Naive Bayes classifier is based on Bayesian classification Performs probabilistic prediction, i.e., predicts class membership probabilities. Suppose there are m classes C1, C2,…, Cm. Given a tuple , X, the classifier will predict that X belongs the class having highest posterior probability, condition on X i.e., the Naive Bayesian classifier predicts that tuple X belongs class C_i if and only if,

$$P(C_i|X=x) > P(C_j|X) \quad \text{for } 1 \le j \le m, j \ne i \qquad (6)$$

$$P(C_i|X=x) = \frac{P(X = x|C_i)P(C_i)}{P(X=x)} \qquad (7)$$

Thus maximize P(Ci| **X=x**). The class C_i for which P(Ci|X=x) is maximized is called the maximum posterior probability P(Ci| **X=x**).

2.10 C4.5 AdaBoost

C4.5 is the decision tree classifier which is embedded with AdaBoost [19]. C4.5 and AdaBoost are two learning algorithms which take a finite training sample S of m labelled examples as input [6].

$$S = \{(xi\ f(x_i))\}_{i=1}^{m} \qquad (8)$$

The x_i are points in some instance space X, and f is the Boolean target function over X. The goal of these algorithms is to find a function with small training error on S in as few "rounds" as possible [6].

Classifier AdaBoost is used to predict the reduction orders for bug data set [19]. There are different methods exist to build decision trees, but all of them summarize given training data in a tree formation, with each division representing an association between feature value and a class label.

3. EXPERIMENTAL RESULTS AND DISCUSSION

The retrieval performance of the proposed system is compared with text classification technique i.e., Naive Bayes in order to measure the efficiency of the proposed system. For evaluation purpose, the data set is classified to test and trained set in order to measure the results of proposed system.

3.1 Calculation of Attributes Value

Table.1 present an overview of all the attributes of the bug data set. For given a bug data set, all these attributes are extracted to determine the features of the bug data set. Among the attributes in Table.1 four attributes are directly calculated from a bug data set, i.e., D1,D4,B1 and D2; six attributes are calculated based on the words in the bug data set, i.e., D2,D3,D5, D6,B3 and B4; five attributes are calculated as the entropy of an enumeration value to indicate the distributions of items in bug reports,ie.e.,B6,B7,B8,B9 and B10; three attributes are calculated according to the further statics, i.e., D7, D8 and B5. All the 18 attributes [18] in Table.1 can be obtained by direct extraction or automatic calculation.

From Table.1, if the prediction result is 0 then the order to reduce the data is FS→IS, otherwise if the prediction result is 1 then the order to reduce the data is IS→FS. The trained and test data set (training set) gives the same predicted result.

Table.1 Calculation of Attributes value in trained data set

Sl. No	Attribute Name	Training Data Set	(Test) Training Data set
B1	# Bug reports	38.0	38.0
B2	# Words	1454.0	1454.0
B3	Length of bug reports	36.82051	36.82051
B4	# Unique words	14.0	14.0
B5	Ratio of sparseness	14.97435	14.97435
B6	Entropy of severities	0.0	0.0
B7	Entropy of priorities	0.0	0.0
B8	Entropy of products	0.0	0.0
B9	Entropy of components	0.0	0.0
B10	Entropy of words	2.639057	2.639057
D1	Fixers	38.0	38.0
D2	Bug reports per fixer	1.000002	1.000002
D3	Words per fixer	703.0	703.0
D4	Reporters	0.0	0.0
D5	Bug reports per reporter	1.0	1.0
D6	Words per reporter	259.0	259.0
D7	Bug reports by top 10 percent reporters	1.157894	1.157894
D8	Similarity between fixers and reporters	-0.30012	-0.30012

Predicted result to reduce the order is FS→IS.

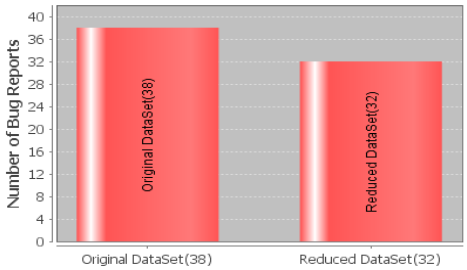

Fig.6 Comparison result between original and reduced bug data set (Training bug data Set)

Fig.6 shows the number of bug reports in original data set (38) and reduced data set (32).By using reduction algorithm, noisy bug reports, uninformative bug reports and bug words get reduced in training set in order to improve the accuracy of bug triage.

From Table.2, if the prediction result is 0 then the order to reduce the data is FS→IS, otherwise if the prediction result is 1 then the order to reduce the data is IS→FS. The trained and test data set (test set) gives the same predicted result.

Table.2 Calculation of Attributes value in Test Data Set

Sl. No	Attribute Name	(Train) Test Data Set	Test data set
B1	# Bug reports	18.0	18.0
B2	# Words	771.0	771.0
B3	Length of bug reports	42.83333	42.83333
B4	# Unique words	17.0	17.0
B5	Ratio of sparseness	17.72222	17.72222
B6	Entropy of severities	0.0	0.0
B7	Entropy of priorities	0.0	0.0
B8	Entropy of products	0.0	0.0
B9	Entropy of components	0.0	0.0
B10	Entropy of words	2.833213	2.833213
D1	Fixers	18.0	18.0
D2	Bug reports per fixer	1.000002	1.000002
D3	Words per fixer	378.0	378.0
D4	Reporters	0.0	0.0
D5	Bug reports per reporter	1.0	1.0
D6	Words per reporter	126.0	126.0
D7	Bug reports by top 10 percent reporters	1.333333	1.3333333
D8	Similarity between fixers and reporters	0.147001	0.147001

Predicted result to reduce the order is IS→FS

Fig.2 shows the number of bug reports in original data set (18) and reduced data set (11).By using reduction algorithm, noisy bug reports, uninformative bug reports and bug words get reduced in training set in order to improve the accuracy of bug triage.

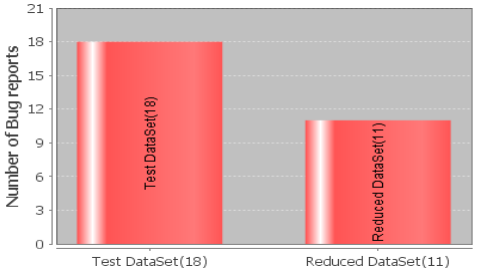

Fig.7 Comparison result between original and reduced bug data set (Testing bug data Set)

Fig.7 illustrates the performance value of precision (0.667), recall (0.737) and F-measure (0.70) for training bug data set.

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

$$Accuracy_k = \#\text{ correct relevant developers} / \#\text{ all data sets} \quad (9)$$

The Presicion and recall value for the trained data set can be calculated by using the formulaes

$$Precision_k = \#\text{ correct relevant developers} / \#\text{ Relevant developers X k} \quad (10)$$

$$Recall_k = \#\text{ correct relevant developers} / \#\text{ Correct developers} \quad (11)$$

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

$$F_k = 2 \text{ X precision X recall} / \text{precision + recall} \quad (12)$$

where k denotes the size of recommendation list.

Table.3 Accuracy and Error rate Table

Sl.No	Name of the Algorithm	Accuracy Rate	Error Rate
1	Naive Bayes	0.786	0.213

From Table.3, the accuracy and error rate are calculated for training bug data set. The accuracy of Naive Bayes algorithm is 0.786 and error rate is 0.213.

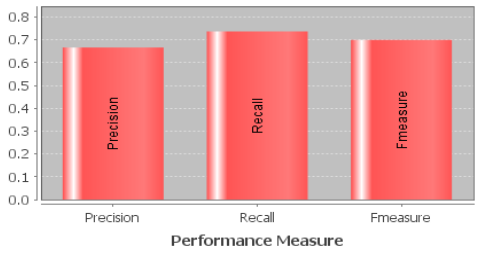

Fig.8 Comparison Graph for Precision, Recall and F-measure

Fig.8 illustrates the performance value of precision (0.667), recall (0.737) and F-measure (0.70) for training bug data set.

Table.4 Performance Values of Precision, Recall and F-measure

Sl. No	Name of the Algorithm	Precision	Recall	F-measure
1	Naive Bayes	0.667	0.737	0.70

From Table.4, the performance values of Precision, Recall are calculated for training bug data set. To balance the precision and recall value, the F measure value is calculated for training bug data set.

4. CONCLUSION AND FUTURE WORK

Bug triage is a costly step of software maintenance in both labor cost and time cost. The proposed system combines the feature selection algorithm (FS) with instance selection algorithm (IS) in order to reduce the scale of bug data sets as well as improve the data quality. A Predictive model is used to determine the order of applying reduction order, i.e., FS to IS or IS to FS. 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. For predicting reduction orders, plan to pay efforts to identify the potential relationship among the attributes of bug data sets and the reduction orders.

5. 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. 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] Bugzilla, (2015). [Online].Avaialble: http://bugzilla.org/

[4] 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.

[5] N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, 2nd ed. Boston, MA, USA: PWS Publishing, 1998.

[6] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in Proc. 13th Int. Conf. Mach. Learn., Jul. 1996, pp. 148– 156.

[7] 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.

[8] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triagewith tossing graphs," in Proc. Joint Meeting 12th Eur. Softw. Eng. Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng., Aug. 2009, pp. 111–120.

[9] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise indefect prediction," in Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng., May 2010, pp. 481–490.

[10] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reportsusing a vocabulary-based expertise model of developers," in Proc. 6th Int. Working Conf. Mining Softw. Repositories, May 2009, pp. 131–140.

[11] Mozilla. (2015). [Online]. Available: http://mozilla.org/

[12] E.Murphy-Hill, T. Zimmermann, C. Bird, and N. Nagappan, "The design of bug fixes," in Proc. Int. Conf. Softw. Eng., 2013, pp. 332– 341.

[13] M. Rogati and Y. Yang, "High-performing feature selection fortext classification," in Proc. 11th Int. Conf. Inform.Knowl.Manag., Nov. 2002, pp. 659–661.

[14] 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.

[15] 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.

[16] 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.

[17] D. R. Wilson and T. R. Mart_ınez, "Reduction techniques forinstance-based learning algorithms," Mach. Learn., vol. 38,pp. 257–286, 2000.

[18] T. Xie, S. Thummalapenta, D. Lo, and C. Liu, "Data mining forsoftware engineering," Comput., vol. 42, no. 8, pp. 55–62, Aug. 2009.

[19] J. Xuan, H. Jiang, Y. Hu, Z. Ren, Z. Luo, W.Zouand X. Wu, "Towards Effective Bug Triage with SoftwareData Reduction Techniques" in IEEE Trans. on Knowl. and Data Eng., vol. 27, no. 1, Jan. 2015.

[20] 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.

[21] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Automatic bug triage using semi-supervised text classification," in Proc. 22nd Int. Conf. Softw. Eng. Knowl. Eng., Jul. 2010, pp. 209–214.

[22] 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.

[23] 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.