International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Improving Auto Bug Triage by Effective Data Reduction

Roshna V. Sangle[1], Rajendra D. Gawali [2]

[1, 2]Department of Computer Engineering, Lokmanya Tilak College of Engineering, Koparkhairane, India

Abstract: ***Large open source software projects like Mozilla, Firefox, Eclipse etc. receive huge number of submitted bug reports daily basis. Manual Triaging of these upcoming reports is error-prone and time consuming process. The purpose of bug triaging is to assign potentially experienced developers to upcoming bug reports. Thus to reduce cost and speedup bug triaging, this paper presents an automatic approach to predict a developer with approximate time required to solve the upcoming report. In proposed system data set reduction is achieved through techniques like stemming, stop word removal, Instance selection and Feature selection on bug data set, which improve the scale and quality of bug data. The simultaneous usage of instance selection and feature selection reduces the scales on bug dimension and word dimension which improves the accuracy of bug triage. The combination of feature selection algorithm, statistics (CHI^2) and instance selection algorithm, Iterative Case Filter (ICF) is applied in proposed paper. Then Naive Bayesian classifier is used to predict the expert developer to fix the upcoming bug. This paper also focuses on how to assign any upcoming bug to new developer whose bug fixing history is not available in training dataset.***

Keywords: Bug Triage,Stemming,Stopwords,Instance selection, Feature Selection,Training dataset,cold-soft etc.

1. Introduction

In Software Development Life Cycle (SDLC) testing is most crucial phase. Quality of software product is mostly depend on testing of a product. Hence Bug triage is most essential process in SDLC. Auto Bug Triage is process of assigning potential developer to fix or resolve the new bug. In an software industry a wrong bug triage can lead to huge loss of money, time & employees efforts.

Traditional software analysis is not completely suitable for the large-scale and complex data in software repositories. Data mining has emerged to handle software data. By leveraging data mining techniques, mining software repositories can uncover interesting information in software repositories and solve real- world software problems. A bug repository plays an important role in managing software bugs. Software bugs are unavoidable and fixing bug is an expensive process in software development life cycle.

Software companies spend over 45 percent of cost in fixing bugs [12]. Large software projects deploy bug repositories (also called bug tracking systems) to support information collection and to assist developers to handle bugs [13]. In a bug repository, 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. A bug repository provides a data platform to support many types of tasks on bugs, e.g., fault prediction [14], bug localization [15], and reopened bug analysis.

The proposed system introduces a new way of achieving efficient auto bug triage over the manual bug triage which is time consuming process. System uses various techniques like stemming, stop word removal, instance algorithm & feature algorithm etc to improve bug triage process. Proposed system does not directly assign this bug to developer to solve it, rather system suggest or recommend the name of developer who can solve this bug efficiently. Instance selection and feature selection algorithms are used to generate a reduced training bug data set. It means original huge dataset is replaced with the reduced data set for bug triage. Instance selection is used to obtain a subset of relevant instances (i.e., bug reports in bug data) while feature selection aims to obtain a subset of relevant features (i.e., words in bug data). Finally text classification algorithm like Naive Bayesian can be used to predict the name of developer who can fix the upcoming bug.

Bug triaging is an essential part of developing software. Based on features of the bug report, such as the title, priority, severity, and affected components, developers have to assess whether a bug report is meaningful and identify a developer most suited for fixing the bug or implementing the required enhancement [4], [16]. This task of identifying potential experts for addressing bug reports is known to be time-consuming, tedious and error-prone, in particular due to the size and complexity of software projects and teams [17]. In the last few years, a variety of research approaches has been developed to automatically support bug triaging by recommending expert developers for bug reports. These approaches mainly differ in the way they identify the expert developers. The one kind of approaches focuses on textual similarity of the bug reports and bases on the assumption that bug reports that are similar in their textual characteristic should be fixed by the same developers. In some approaches, the term bug reports also encompasses tasks, issues and other work items. focus solely on discovering textual similarities between bug reports and use machine learning and information retrieval without looking at the code (e.g. [4]).

Bug Life Cycle:
Generally bug passes through many states during its life time as shown in following figure 1.

Licensed Under Creative Commons Attribution CC BY

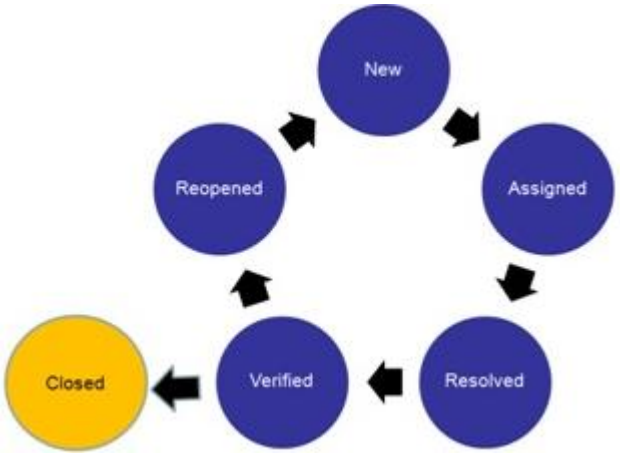

Figure 1: Bug Life Cycle

Sample Bug Report:

Figure 2: Mozilla Bug Report

2. Related Work

Bug Triaging typically involves two activities in open source projects. First developers check whether the bug has been already reported ,i.e., is it a duplicate of another existing bug report? If it is not, then it is assigned to a developer who is then responsible to fix the bug. Many approaches have been proposed to automate these steps.

To detect duplicates, several approaches use natural language processing (NLP) techniques on the bug description [5,6]. D. Čubranić et al, proposes very first approach to perform bug triage automatically. They used Naive Bayesian classifier for prediction of developer[1].Further their work is extended by john Anvik et al, they used a semi-automated approach. they used a support vector machine (SVM) classifier for prediction[2].In the year 2009 ,J. Xuan et al, proposed a semi-supervised approach, where they used combination of clustering and classification. for clustering they have used Expected Maximization (EM) algorithm whereas for classification they used Naive Bayesian algorithm[3]. They used Eclipse dataset for their work. Syed Nadeem et al,[4] proposed a methodology which uses Weka Tool for their implementaion. They have examined their work with 7 different classification algorithm like J-48,Decision tree, SVM, NB classifer etc. They found that Support Vector Machine (SVM) has highest accuracy & J-48 gives lowest accuracy among all seven algorithms. In this reduction of feature is done by two methods, one is based on feature's global frequency thresholding and second is Least Semantic Indexing (LSI). Further Dominique et al ,[6] proposed a vocabulary based model , in which they compares vocabulary found in source code & bug report created by same developer. If maximum similarity found then that bug is assign to that developer. For this they have used "Diff" command to check the similarity. For this they generated a Term-author matrix. Tao Zhang et al, in the year 2014 proposed a bug triage methodology based on Topic model and developer relations. In which they used Topic Modeling Toolbox (TMT) to find the words .They also performed preprocessing on bug reports by tokenization, stop word removal & stemming. They used Laent Dritchlet Allocation (LDA) to extract topics from historical bugs. In 2015 Jifeng Xuan et al,[7] proposed a novel approach which aim on training dataset reduction in order to enhance the performance of their previous work[3].They used 4 different Instance Selection & 4 Feature Selection algorithm to find the best IS & FS algorithm ,which gives better reduction of training dataset. They also checked reduction of dataset in two different way first by IS-FS as well as FS-IS and they found that order of FS-IS gives the better result. Naive Bayesian Algorithm is used for prediction of developer by them.

3. Methodology

This section gives overview of how will system will work in order to achieve efficient Auto Bug Triage.

3.1 Data Processing

This step initially removes noisy & irrelevant data from bug reports, using stemming & stop word removal. Stemming is done in order to bring different keywords in their base form. e.g. "computes", "computation", "computing", "computed" are brought to their base form "Compute" and stop word removal removes the unnecessary or least useful words like "a", "an"', "the", "to", "of", "for" etc. from bug reports .

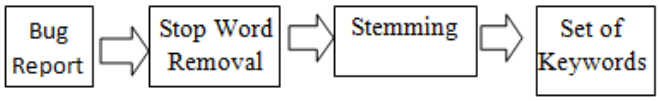

Figure 3: Bug Report Preprocessing

3.2 Bug data Reduction

It combines the existing techniques [7] of instance selection and feature selection to remove certain bug reports and words. Thus reducing the Data Scale i.e. Bug Dimension & Word Dimension which will help to improve the accuracy to Predict developer to solve the bug.

Licensed Under Creative Commons Attribution CC BY

3.3 Prediction of developer to Fix the Bug

Prediction of a developer who can fix the.bug using NB classifier. Naive Bayesian classifier perform reduction of training dataset, which will extract different attributes to describe each bug data set. Such attributes can be extracted before new bugs are triaged.

3.4 System Approach

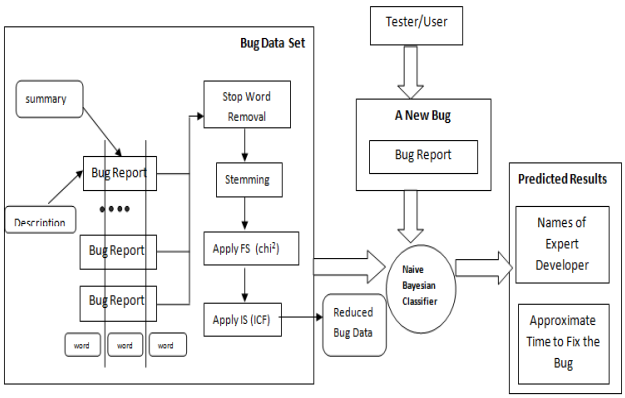

Figure 4: System Architecture

3.5 Dataset

As shown in table 1system uses following Mozilla dataset as a training dataset. It is retrieved from Mozilla bug repository in the form of JSON array[19].

Table 1: Dataset Details

Sr. No.	*Mozilla Dataset*
1	DS_M1_Mozilla_400001 To 440000
2	DS_M2_Mozilla_440001 To 480000
3	DS_M3_Mozilla_480001 To 520000
4	DS_M4_Mozilla_520001 To 560000
5	DS_M5_Mozilla_560001 To 600000

3.6 Usability Metrics

Followings are the usability metrics which will evaluate the systems performance in terms of how much redundant or noisy reports are removed after applying instance selection algorithm i.e. Iterative Case Filter on above mentioned training dataset.

Table 2: Usability Metrics

Metric	Definition
Recall	{All Reports} ∩ { Duplicate Reports} / { Duplicate Reports}
Precision	{All Reports} ∩ { Duplicate Reports} / { All Reports }
F-Measure	FMeasure= 2 {Precision} ×{Recall} / {Precision +Recall}

4. Algorithms

4.1 Chi² Algorithm

The Chi² algorithm applies the X^2 statistic test which conducts a significance test on the relationship between developer and their respective keywords present in that Bug report . It consists of two phases. In the first phase, it begins with a large significance level (a), e.g., 0.5,for each keyword in bug report attribute, the following is performed:1) calculate the CHI² value as in (1) for every pair of adjacent intervals (at the beginning, the number of intervals equals the number of distinct values of an attribute); 2) merge the pair of adjacent intervals with the lowest X^2 value being the critical value.

Formula for calculating CHI² Score is given as in eq (1)

$$\chi^2 = \sum_{allcells} \frac{(Oij - Eij)^2}{Eij} \quad (1)$$

where Oij is the observed frequency and Eij is the expected (theoretical) frequency, asserted by the null hypothesis. The greater the value of χ2, the greater the evidence against the hypothesis H0 is. Here our Hypothesis is that "**keywords present in upcoming bug reports are unique**". It means if system gives CHI² score greater than significance level i.e. 0.5 then it conclude that keywords extracted from upcoming bug report are not unique.

4.2 Iterative Case Filter Algorithm

Instance selection is algorithm used to lessen the number of instances i.e. Bug Reports and to enhance the training set quality. According to [8], Iterative Case Filter (ICF) [9] is chosen as the instance selection algorithm in this work. ICF is an instance selection algorithm based on the k-Nearest Neighbour algorithm (KNN) [10].

4.3 Naive Bayesian Classifier

It is based on Baye's rule.This classifier work efficiently with nominal dataset as well as efficiently handles large data. It assumes that all attributes as independent.

The Baye's rule is given as follows in equation (2):
Given a hypothesis h and data D bears on the hypothesis:

$$P(h/D) = \frac{P(D/h)P(h)}{P(d)} \quad (2)$$

where p(h) is Prior probability hypothesis h ,(D/h) is the maximum likelihood and P(h/D) is the posterior probability.

5. Result

Following table 3 shows various results that system obtained in terms of recall ,precision & FMeasure.

Table 3: Analysis Result

Sr. No.	Parameters	Value
1	Recall	13.64
2	Precision	86.364
3	FMeasure	53.801

6. Conclusion

Bug triage is an essential and expensive stage of software development cycle, in terms of both labor cost and time cost. System's objective is to provide a improved techniques to do

Licensed Under Creative Commons Attribution CC BY

auto bug triage efficiently. The results shows good bug data set reduction which speedup the bug triage process .system also gives approximate time required for developer to fix the bug which is useful to decide to whom bug must be assigned as per its priority and severity. system also able to assign the upcoming bug to any new developer whose past bug fixing history is not available.

References

[1] D. Čubranić and G. C. Murphy, "Automatic bug triage using text categorization," Proc. Intl. Conf. Software Engineering & Knowledge Engineering (SEKE 04), Jun. 2004, pp. 92-97.
[2] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug?.In Proceedings of the 28th international Conference on Software Engineering (Shanghai, China, May 20-28, 2006). ICSE06. ACM, New York, NY, 361-370.
[3] 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.
[4] Syed Nadeem Ahsan, Javed Ferzund, Franz Wotawa "Automatic Software Bug Triage System(BTS) Based on Latent Symentic Indexing and Support Vector Machine",2009 IEEE.
[5] Tao Zhang, Geunseok Yang,Byungjeong Lee,eng Keong Lua," A Novel Developer Ranking Algorithm For Automatic Bug Triage Using Topic Model And Developer Relations", 2014,IEEE.
[6] Dominique Matter,Adrian Kuhn,Oscar Nierstrasz "Assigning Bug Reports Using A Vocabulary-Based Expertise Model Of Developers",2009 IEEE.
[7] Jifeng Xuan,He Jiang,Yan Hu,Zhilei Ren, Weiqin Zou,Zhongxuan Luo,Xindong Wu "Towards Effective Bug Triage with Software Data Reduction Techniques".vol. 27, no.1,Janury 2015.
[8] M. Grochowski and N. Jankowski, "Comparison of instance selection algorithms II, results and comments," Proc. Intl. Conf. Artificial Intelligence and Soft Computing (ICAISC 04), Springer, Jun. 2004, pp. 580-585.
[9] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," Data mining and knowledge discovery, vol. 6, no. 2, Apr. 2002, pp. 153-172.
[10] T. Mitchell, Machine Learning, McCraw Hill, 1996.
[11] 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.
[12] R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th ed. New York, NY, USA: McGraw-Hill, 2010.
[13] B. Fitzgerald, "The transformation of open source software," MIS Quart., vol. 30, no. 3, pp. 587–598, Sep. 2006.
[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] S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D. Ernst, "Finding bugs in web applications using dynamic test generation and explicit-state model checking," IEEE Softw., vol. 36, no. 4, pp. 474–494, Jul./Aug. 2010.
[16] www.eclipse.org
[17] https://bugzilla.mozilla.org/rest/bug.

Licensed Under Creative Commons Attribution CC BY