Classifying Bug Severity Using Dictionary Based Approach

Shruti Gujral
Department Of CSE
CU, Gharuan, India
shrutigujral.cu@gmail.com

Gitika Sharma
Department Of CSE
CU, Gharuan, India
gitikasharma41@gmail.com

Sumit Sharma
Department Of CSE
CU, Gharuan, India
cu.sumitsharma@gmail.com

Diksha
Department Of CSE
CU, Gharuan, India
dikshanagpal91@gmail.com

***Abstract*— Bug tracking system allows user to report bugs that they encounter while operating the software. These bugs are then received by developers and they resolve these bugs according to their severity level. This task of assigning severity level is manual task that need expertise of assessing the severity level of reported bug. But if these reported bugs are large in number then manual process to assess severity level of bugs becomes very hectic. So there should be an automatic process to classify it so that bugs that need immediate fixation get resolved early. A few attempts have been made by researchers to automate the task. The approach followed in this paper has made an attempt to automate the bug severity classification using text mining technique and Naïve Bayes Multinomial classifier. This paper further proposes an approach of making this task more efficient using dictionary of bug terms.**

Keywords—software bug classification; Bug severity; Bug Tracking system; Dictionary

I. INTRODUCTION

A software bug is commonly used to describe the occurrence of a fault in a software system which results it to act differently from its specification [1]. Bugs occur during testing phase or after deployment of software. Bugs are mostly mistakes which originate from human participation. 57% bug originates from error made by human due to carelessness and absent mindedness [2].Bugs can have different types of ripple effects from inconvenience to user - crash of software. Some bugs have only minor effect on the functionality of program and can be left undetected. But severe bugs can cause the program to crash or freeze and may lead to a denial of service. Other bugs can be qualified as security bugs for example enable a bogus user to allow access controls so that they can obtain unauthorized access. The reason of 2003 North America black out was local outage that was left undetected due to race condition in General Electric Energy XA/21 monitoring software [3] hence presence of bugs can prove quite costly.

Moreover, with the increasing dependence on software systems, software quality is becoming more important. So to ensure quality in software, different methods are used such as code reviews and rigorous testing to remove bugs as early as possible to prevent the loss it may cause. Different bugs have different impact on software quality. Bugs can be classified based on its negative impact or severity on software quality. Bugs are generally classified as critical bugs, major bugs, minor bugs and trivial bugs [4]. Critical bugs are more severe while trivial bugs are just inconvenience to user thus less severe. Software bug classification helps in bug triaging system. Bug triaging are the steps that are taken to manage a bug from the time it is reported to the time the bug is resolved [5]. Effective bug triaging is crucial part of software system as it is evaluation of the reported bug. It ensures that bug is filed in correct place and has enough description. Project Manager facilitates bug triaging meeting with expert member from business sector, development and tester. After bug triaging meeting bugs are classified into three categories according to severity of bugs. These categories are the bugs that will be fixed now, bugs that will be fixed later and bugs that will never be fixed.

Many open source software provide bug tracking system along with them and keeps track of the reported bug by end user. Bugzilla, jira ,Fog Bugz,Fossile and ikiwiki are some example of bug tracking system[6].To report a bug in bug tracking system, a user is asked to fill a form that provide necessary details about the bug. Then this form is used by the development team to resolve the bug. This form has one-line summary of the failures which are observed and also includes detailed description of it. This form also provides the facility of selecting a particular component in which that bug occurs. The numbers of bug reports that are reported in tracking system are generally higher in number so it becomes difficult to resolve all the bugs on time. Therefore developer has to make choice among all reported bug based on severity i.e. more severe bugs should be resolved bugs. But it is very tedious job to manually assign severity. Automation of this work can be helpful from developer's point of view. Bug reports come with textual description, so to classify these reports text mining algorithms can provide such support. In past, text mining techniques have been applied by many researchers. These techniques were applied on the descriptions of bug reports for automation of the bug triaging process [7] and for detection of duplicate bug reports [8].

Our hypothesis is that terms that are used frequently to describe bugs severity should be categorized into severity

978-1-4799-84336/15/$31.00©2015 IEEE

terms and non-severity terms. Dictionary of terms describing severity can be created and incoming reports automatically can be assigned severity based on terms specified in bug report.

The paper itself is structured as follows. Section II provides the related work of attempt to make bug triaging system automatic using text mining and machine learning algorithm. Section III provides methodology used by us to predict bug severity along with the result obtained. Section IV provides discussion and future work. The result of discussion can be used to fully automate the process of bug classification. Finally, Section V summarizes the results and points out future work.

II. RELATED WORK

We have highlighted some related studies of predicting severity level of reported bugs using text mining and machine learning algorithms.However this section is not a complete list of all related work. The first attempt to automate bug triaging was done by Cubranic et al. using text mining approach and naïve bayes algorithm and achieved accuracy 30 % [9]. The same work was extended by Anvik et al., they used some different algorithms for supervised learning and labeling heuristic approach. Precision levels of Eclipse and Firefox datasets was obtained 57% and 64% respectively.

Independent Verification and Validation Facility projects of NASA was used by Menzies et al. to predict the severity of bugs by applying rule learning method on textual descriptions of reported bugs [11], precision level between 0.08 and 0.91 and recall level between 0.59 and 1.00 was achieved. In another attempt of classifying bug severity,seven different machine learning algorithm was applied on bug reports of Mozilla and highest accuracy of 44.4% was obtained using support vector machine and latent semantic accuracy [12].

Another attempt was made for predicting the severity levels based on the description of the newly reported bug report of three open source project Mozilla, eclipse, GNOM to severe and non-severe classes by applying the Naive Bayes classification algorithm[13].The same work was extended for the comparision of machine learning algorithms such as Naive Bayes Multinomial, KNN and SVM for classification of bug severity into severe and non severe class[14].K. K. Chaturvedi et al. made an attempt for automating process of bug classification on basis of severity level of bug report data of NASA' Independent Verification and Validation Facility project from PROMISE repository, various machine learning algorithm Naïve Bayes, k-Nearest Neighbor, Naïve Bayes Multinomial, Support Vector Machine, J48 and RIPPER were used and accuracy was compared[15].Cheng-Zen Yang et al. investigated the influences of four quality indicators of bug reports in severity prediction instead of using textual description alone.Eclipse dataset was used in empirical study and result indicated that these indicators further improved the performance of previous work employed only on textual description [16].

From literature survey we conclude that software bug can cause harmful effects on software. So if a software bug occurs then we need to fix it as soon as possible based on its negative impact on software system, but task of classifying bugs based on its severity is manual process We need expert to examine the reported bugs and assign it severity level so that bugs with more severe effects get fixation early than bug with less severity. Many efforts are taken by different researcher to make this process automatic using machine learning as discussed in our literature survey. But still it is far from giving reliable accuracy of automatic prediction of severity level of a bug.

III. WORK DONE

In our approach we have worked on bug repository of eclipse. The methodology used for performing classification of bugs follow steps which are mentioned and explained as follows.

A. Extraction of data from bug repository -

The experiment considers 1096 bug reports sample of eclipse that was reported in bugzilla [17]. It consists of 586 severe and 511non severe bug reports. Bugzilla has six level of severity trivial, minor, normal, major, and critical and blocker. Bug reports are categorized into severe and non severe classes based on severity level in the experiment. Therefore, we consider the trivial and minor severity levels as non-severe bugs, while major, critical and blocker severity levels are considered as severe bugs. Bug reports of type normal severity is not considered in experiment as it is default option in bugzilla for severity level. It needs manual classification to determine the severity level [18].

B. Applying text mining technique

On textual description of bug reports standard preprocessing steps are performed which include tokenization, stop word removal and stemming.

a) Tokenization: The purpose of tokenization is to explore the words in a sentence by removing punctuation marks and other characters like hyphen and brackets.

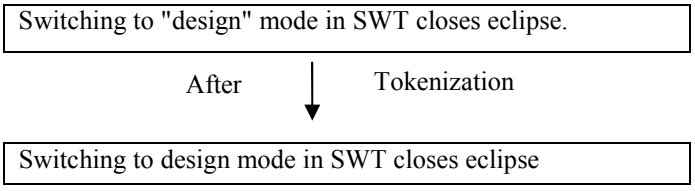
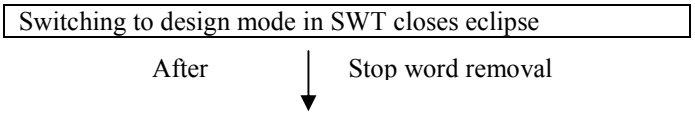


Fig. 1. Tokenization

b) Stop word removal: This process reduces textual data by removing unnecessary words and hence it improves system performance.



Switching design mode SWT closes eclipse

Fig. 2. Stop word removal

c) Stemming: The purpose of this technique is to reduce words into their root words. Porter algorithm has been used to reduce term to its root word.

Switching design mode SWT closes eclipse

After ↓ Stemming

switch design mode swt close eclips

Fig. 3. Stemming

C. Extraction of terms specifying severity level

TF-IDF (term frequency (TF) and inverse document frequency (IDF)) are used for extracting the terms that can be used for categorizing severe and non-severe bugs.

TF = word w in document d/ total word in document d (1)

IDF = log (total document / total document that contain word w) (2)

TF-IDF = TF * IDF (3)

D. Classification using naïve bayes multinomial algorithm

Training and evaluation set of 70%-30% of available reports are chosen for the experiment. Naïve bayes multinomial classifier is trained using training set and then performance is evaluated using evaluation set.

E. Performance measurement

Precision and accuracy measures are used for evaluating the performance of our approach.

a) Precision : Precision is calculated as percentage of bug reports predicted as severe or non-severe that is actually correctly predicted.

Precision= Σ True positive / Σ Test outcome positive (4)

b) Accuracy : Accuracy is calculated as arithmetic mean of precision and inverse precision.

Accuracy= (Σ True positive + Σ True negative)/ Σ Total Population (5)

The figure 4 shows the comparison of accuracy and precision using naïve bayes multinomial classifier. It was found that accuracy rate is 72% and precision is 69%.All the results are obtained using rapid miner tool [18].

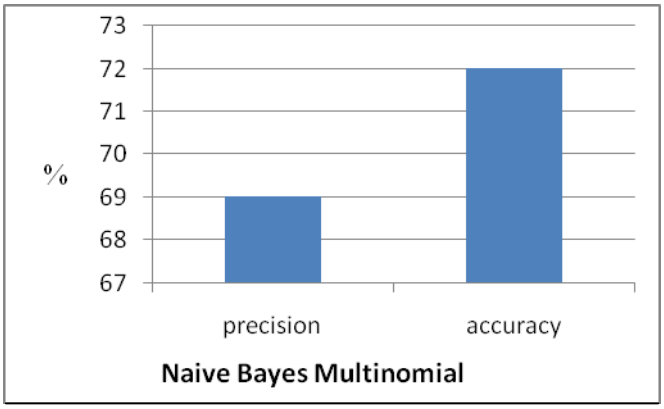


Fig. 4. Accuracy and precision using naïve bayes multinomial

IV. DISCUSSION AND FUTURE WORK

In the methodology mentioned in section III, the method of classifying bugs based on textual description given in the bug report has been implemented. Classifying bug will be helpful from developer's point of view as he will be able to fix the bugs that need immediate fixation. This paper proposes an algorithm of bug severity classification which can further improve the performance of automatic process of bug severity classification. The algorithm includes creating dictionary of critical terms that help in deciding severity level automatically. These critical terms are gathered after preprocessing of bug reports. Some of critical terms of severity level and non severity level are as follows:

TABLE I. CRITICAL TERMS

Critical terms of eclipse bug reports	
Severe terms	adapt , hang, , deadlock, crash, anymor, freez ,thread,
Non severe terms	style, cce, whitespac, runnabl, system, put, param, ,

Proposed_Algorithm_1: Bug severity classification

1. Select the product for bug classification
2. Download the bug file/repository
3. Preprocess the bug file
4. Create dictionary of critical words which specify severity
5. Calculate TF-IDF (Term Frequency-Inverse Document Frequency)
6. Update dictionary with respect to TF-IDF using machine learning

Proposed algorithm for bug severity classification is as follows:

V. CONCLUSION

In this paper bug classification using eclipse bug reports has been performed. Precision and accuracy level of 72% and 69% respectively has been calculated. This paper further proposes an algorithm to improve bug classification by creating dictionary of terms which will be updated using machine learning. The proposed work will be implemented and same work can be performed by selecting a particular component. Local dictionary of particular component can be created and severity level can be predicted.

REFERENCES

[1] R. Patton, Software Testing (2nd Edition). Sams, 2005.

[2] Kiyoh Nakamura, Mamoru Sugawara and Masahiro Toyama "Defect Prevention Activities And Tools"IEEE 1991.

[3] "Software Bug Contributed to Blackout". Retrieved 2008-01-07.

[4] "Defect severity classification in software testing (with an example)", http://www.zyxware.com/articles/3559/defect-severity-classification-in-software-testing-with-an-example, May 24, 2013.

[5] Iftekhar Ahmed, Nitin Mohan and Carlos Jensen "The Impact of Automatic Crash Reports on Bug Triaging and Development in Mozilla"ACM 2014.

[6] "15 Most Popular Bug Tracking Software to Ease Your Defect Management Process", http://www.softwaretestinghelp.com/popular-bug-tracking-software/".

[7] MamdouhAlenezi and Kenneth MagelShadiBanitaan "Efficient Bug Triaging Using Text Mining" ACADEMY PUBLISHER 2013.

[8] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in Proceedings of the 29th internationalconference on Software Engineering, 2007.

[9] D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," in Proc Sixteenth International Conference on Software Engineering, Citeseer, 2004, pp.92–97.

[10] J. Anvik, L. Hiew, and G. Murphy, "Who should fix thisbug?" in Proc 28th International Conference on SoftwareEngineering. ACM, 2006, pp. 361–370.

[11] T.Menzies and A. Marcus,"Automated severity assessment of software defect reports," in IEEE International Conference on Software Maintenance, 28 2008-Oct. 4 2008, pp. 346–355.

[12] Ahsan, S.N, Ferzund, J. ,Wotawa, F. "Automatic Software Bug Triage System (BTS) Based on Latent Semantic Indexing and Support Vector Machine"in Software Engineering Advances, 2009. ICSEA '09.

[13] A. Lamkanfi, S. Demeyer, E. Giger and B. Goethals. Predicting the severity of a reported bug. In Mining Software Repositories (MSR) 2010: 1-10.

[14] A. Lamkanfi, S. Demeyer, Q.D. Soetens and T. Verdonck. Comparing mining algorithms for predicting the severity of a reported bug. CSMR 2011: 249-258.

[15] Chaturvedi, K.K, Singh, V.B.," Determining bug severity using machine learning technique"in Software Engineering (CONSEG), 2012 CSI.

[16] Cheng-Zen Yang, Kun-Yu Chen,Wei-Chen Kao.Chih-Chuan Yang "Improving Severity Prediction on Software Bug Reports using Quality Indicators" in Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference.

[17] Bugzilla: https://bugs.eclipse.org/bugs/ [Last Access-1/21/2015].

[18] Rapid miner tool: https://rapidminer.com/ [Last Access- 1/21/2015].