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: 5.258

IJCSMC, Vol. 5, Issue. 7, July 2016, pg.523 – 528

A Systematic Approach for Bug Severity Classification using Machine Learning's Text Mining Techniques

Prabhsharan Kaur[1], Charanjeet Singh[2]

[1]Assistant Professor CSE Dept, NWIET, Dhudike, India
[2]M.Tech Scholar CSE Dept, NWIET, Dhudike, India
[1] prabhsharanbrar@hotmail.com, [2] charansheenh77@gmail.com

ABSTRACT- ***In this research study an approach of creating dictionary of critical terms is used to assess the bug severity as severe and non severe. It is found that using different approaches of feature selection and classifier the pattern of accuracy and precision is approximately same. However Chi square test and KNN classifier give the maximum performance of precision and accuracy for the all four components. This research work helps trigger in classifying bugs based on severity and assigning these bugs to relevant developer.***

Keywords- KNN, NBM, TDM, Chi Square, Bug Severity.

I. INTRODUCTION

With the increasing dependence on software systems, importance of software quality is becoming more critical. There are different ways to ensure quality in software such as code reviews and rigorous testing so that bugs can be removed as early as possible to prevent the loss it may cause. There is an old saying, "Every software program is never perfect, there is always at least one bug in it which can be encountered at any time."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]. It is encountered while operating the product either under test or while in use. Bugs are mostly mistakes which originate due to human participation. 57% bug originates from error made by human, which could be either due to carelessness or absent mindedness [2]. When they lead to software failure these bugs can cost companies a big amount of money and in some case loss of human lives e.g. software bug in the a Royal Air Force Chinook aircraft's engine control computer caused it to crash in the year 1994 and 29 people were killed[3]. Therefore early detection of bugs and their resolution was very critical.

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 [6].Effective bug triaging is very important to any

software system. It is evaluation of the reported bug. It involves making sure that bug has enough description to make it easily understood by the developer.

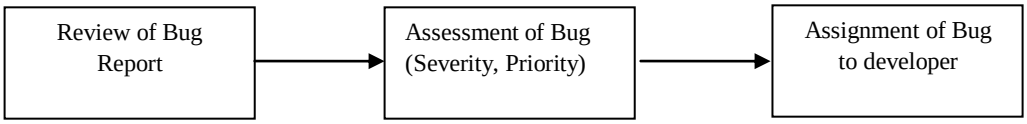

Fig.1: Bug Triaging Process

Project Manager facilitates bug triaging meeting with expert member from business sector, development and tester. In this meeting it is decided that which bug will be fixed and to whom this will be assigned for fix and which bug will be fixed later or will be left undetected. Software repositories contain important information about software projects. It is vital database in modern software development. Many software projects create and maintain bug repositories for storing and updating information of problem [7]. This information can facilitate to manage the improvement of these projects. In the last decade, practitioners have analyzed and mined these software repositories to support software development and evolution. Bug tracking systems are one of the important repositories among all available software repositories. Many open source software projects have an open bug repository that allows both developers and users to submit defects or issues in the software, suggest possible enhancements, and comment on existing bug reports. One potential advantage of an open bug repository is that it may allow more bugs to be identified and solved, improving the quality of the software produced [8].

II. RELATED WORK

Davor Cubanic, Gail C. Murphy [1] made first attempt in 2004 and is proposed to apply supervised machine learning algorithms to assists in bug triaging task. The approach predicts the developer to which the reported bug should be assigned. The work was applied on 15,859 bug reports of Eclipse project. Text mining algorithms and Naïve bayes was used in the approach. The accuracy level 30 % was achieved.

John Anvik et al. [2] extended the work of Davor Cubanic. The authors used some different algorithms for supervised learning such as Support Vector Machine (SVM), Naïve Bayes and C4.5. The approach was used to generate recommendation of developer to assign new bug report. This approach was applied on bug reports of Eclipse, Firefox and gcc (Compiler). Precision levels of Eclipse and Firefox datasets was obtained 57% and 64% respectively with SVM but for gcc there was less positive result.

John Anvik [3] further extended his previous work [3] and created a recommender for assigning the bug reports. The recommender for a project is constructed by using recommendation algorithm. That algorithm used information of the bug reports that was fixed in past and create a model of expertise of developers of project. The recommender thus formed is used to offer the set of recommendations of developers for assigning new bug reports.

Gaeul Jeong et al. [4] coined a new term called "tossed" (reassignment). Bug tossing signifies the reassignment of bug report to new developer. Authors introduced a tossed graph model using Markov chains. The experiment was performed on 450,000 bug reports of Eclipse and Mozilla. The result indicated that bug tossing activity was reduced to 72 %. Thus prediction accuracy was increased up to 23 % as compared to existing approach.

Israel Herraiz et al. [5] analyzed the bug reports of Eclipse. It was concluded that these bug report has too many options for severity and priority field. It was hard from user's point of view to distinguish them. The authors recommended making the bug report simpler than existing format. Severity levels could be reduced to three levels as important, non important and request for enhancement based on time taken to close the bug. Similarly priority field in bug reports were grouped according to mean time taken to close the bug. It was found that this field could also be classified into three levels i.e. high, medium, low.

Tim Menzies et al. [6] in their paper proposed a new automated method called Severity issue assessment (SEVERIS) to assign severity level. It was based on text mining approach and machine learning techniques. Authors conducted a case study using data sets of bug report of NASA's Project and Issue Tracking System (PITS) and

applied SEVERIS on it. The result of case study indicated that SEVERIS was a good method of predicting issue severity levels.

Giuliano Antoniol et al. [7] presented an approach to create an automatic routing system that routed the real bug to maintenance team and request for enhancement to the team leader automatically. This approach considered 1800 issues from BTS of Eclipse, Mozilla and JBoss. Text mining technique was applied on description of report. The classifier was build using three supervised ML techniques like naïve bayes, decision trees and logistic regression. The performance of this approach was evaluated. It indicated that recall and precision level of Eclipse, Mozilla and JBoss was obtained between 33 % to 97 % and 64 % to 98% respectively.

Syed Nadeem et al. [8] suggested an approach to automate bug triage system that predicts the developer to bug reports. Bug reports sample of 1983 of Mozilla were used and feature selection and feature reduction method was applied on them. Then seven different machine learning methods were used for classification. The best result was obtained using latent semantic and SVM and 44.4 % accuracy level was achieved. The value of recall and precision of approach was 28 % and 30 % respectively.

Ahmed Lamkanfi et al. [9] proposed a new method for classifying bugs based on severity. Bug reports of Eclipse, GNOME and Mozilla were preprocessed using text mining algorithms (tokenization, stop word removal, stemming). Then machine learning classifier naïve bayes was applied. The average precision and recall of Eclipse and Mozilla was 0.65-0.75 respectively and 0.70-0.85 in case of GNOME.

Thomas Zimmerman et al. [10] conducted a survey among developers and user of APACHE, Mozilla and Eclipse. The result of survey indicated that there was mismatch between the information required by developer to the information provided by user. Further to overcome this mismatch authors define a new tool CUEZILLA that assess the quality of new bug report. This tool also provided the recommendation about the elements that should be added in bug report to improve the quality of it. The tool was trained on 289 bug reports samples and it calculated the quality of bug report 31- 48 % accurately.

III. PROBLEM FORMULATION

From the extensive literature survey, it is concluded that early detection and classification of bugs is very critical for maintaining the quality of software**.**Bug tracking system helps developer and user to report the encountered bugs in open source software. These reported bugs need to be classified based on severity level as it helps to decide which bug need immediate fixation among all reported bugs. However it is manual task and depends on expert's knowledge. In case of larger number of bug reports, it takes lot of time to classify bug reports.Therefore, there is need for designing a system which would automatically classify the bug reports as severe or non-severe based on their textual description. Although, many attempts has been made at the problem and several machine learning and text mining algorithms have been applied for the same, a novel idea of using dictionary of critical terms (terms specifying severity level) for the prediction of severity level of reports is used in this research. KNN and Naïve Bayes are being exploited as classifiers with information gain and Chi square. Thus, the problem statement for the research is "Bug Severity Classification using Machine learning".

IV. METHODOLGY

Bug reports are extracted from respective bug repository. Then preprocessing on textual information of bug reports is applied to obtain more reliable information. Term-document matrix (TDM) is created and by using feature selection method dictionary of critical terms is created. Then reduced TDM obtained by using critical dictionary terms is fed to classifier for classification of severe and non severe bug report.

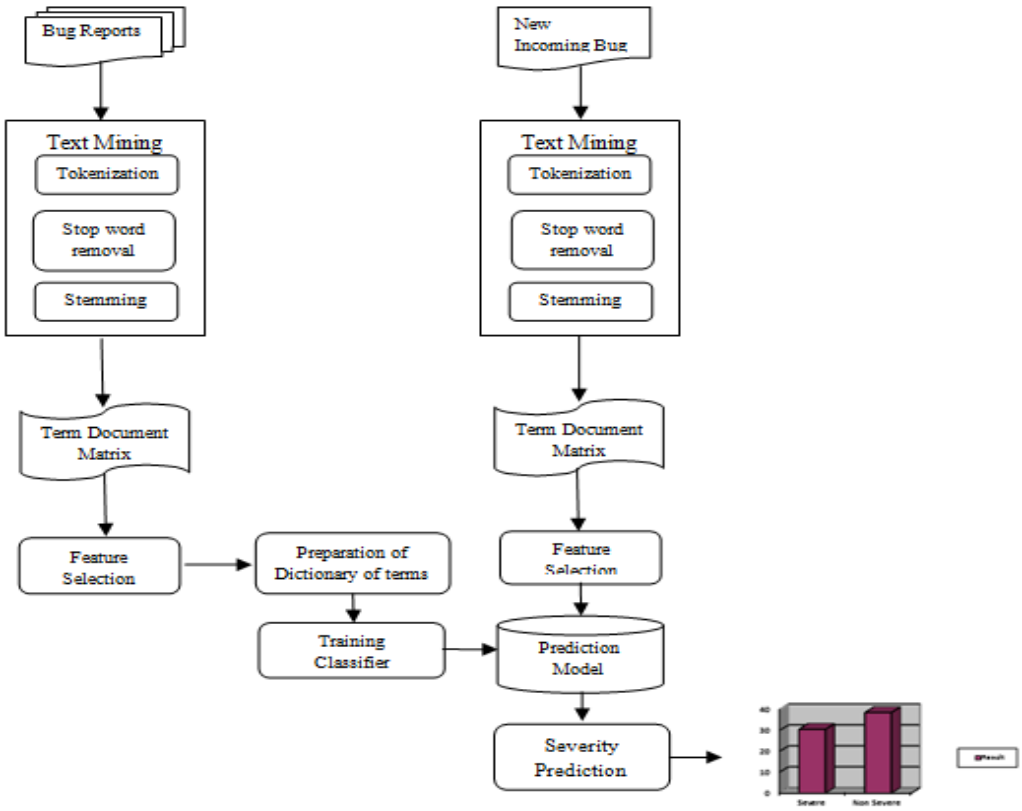

Fig.2: Detailed Methodology

ALGORITHM_NAIVE BAYES MULTINOMIAL

1. Count the distinct terms called vocabulary terms V and number of total terms N.

2. For each c ε C
   Count documents of class c as $N_{c.}$
   Calculateprior probability of document of class c.
   Count number of occurrence of term t i.e. $T_{ct.}$
   for each t ε V
   Calculate conditional probability$P(t_m/c) = \frac{T_{ct}+1}{\sum_{t' \in V} T_{ct'}+B}$
   end for
3. For each c ε C
   Calculate$P\left(\frac{c}{x}\right) = p(c)\prod_{1 \le m \le n_x} P(t_m/c)$
   end for

4. Testing document is labeled class that has maximum value of P(c/x)

KNN ALGORITHM

1. Compute the distance d from training points (a1, a2, a3….aN) to testing point (t_x) using distance function.
2. Sort the training points according to distance from the training point in ascending order.
3. Top k training points are chosen as nearest neighbor to training points.
4. Most common class level c of k training points is assigned to test point.

V. RESULTS

In this research, component specific dictionaries are created of four component of Eclipse. These dictionaries are created using top 125 terms using two feature selection methods; namely-info-gain and Chi square. The set of dictionary terms are then fed to two widely used ML algorithms named Naïve Bayes and KNN for classification task and performance is analyzed in terms of precision and accuracy.

Probability based classification:

(a) Preparation of dictionary using Information gain– The results obtained after using classifier and info gain is shown in table 1. The accuracy varies form 69 % to 75 %, while precision of severe and non-severe category is found to be varying from 65-69 % and 77-84% respectively. The reason of less precision of the severe category than the non-severe ones could be attributed to the fact that the terms extracted by info gain has more bits of information than the severe class level in the datasets used in our experiments. In this approach of creating dictionary using info-gain and NBM classifier, the performance of UI component has been found to be the best.

Table 1: Results of info-gain and Naïve Bayes multinomial

Component	Accuracy	Precision (Severe)	Precision (Non-Severe)
SWT	69.67%	66.77%	78.03%

(b) Preparation of dictionary using Chi square test- The performance of four datasets after using Chi square method and naïve bayes multinomial classifier is shown below in table 5.2. The accuracy of this experiment lie between 68 % and 74 %. While precision of severe class ranges from 65 % to 70 %, the precision of the non-severe class is found to be between 77 % to 84 %.The similar pattern of performance is achieved in these datasets using Chi square as achieved in information gain. SWT component has least accuracy and UI component achieve more accuracy.

Similarity based classification:

(a) Preparation of dictionary using Information gain- The result is obtained using dictionary formed by Chi square test and KNN classifier and shown below in table 2. The accuracy ranges from 87 % to 91 %. The precision of severe class and non-severe class is obtained as lying from 91 % to 96 % and 79 % to 90 % respectively. UI achieves maximum accuracy of 91 % using nearest neighbor classifier.

Table 2: Results of info-gain and KNN

Component	Accuracy	Precision (Severe)	Precision (Non-Severe)
SWT	87.51%	96.97%	79.62%

(b) Preparation of dictionary using Chi square test- The performance of classification task after using Chi square method. Hence, it can be concluded from the results that KNN performs better for bug classification severity under the given experimental set up. The probable reason behind better performance of KNN can be attributed to the fact that useful preprocessing steps were applied to raw dataset extracted from, Bugzilla repository. These preprocessing steps and the process of dictionary formation further refined the datasets. This enable KNN to efficiently classify to bug to their respective classes, as it is well known fact that KNN performs best when noise free data is fed to it. Therefore, it has been concluded that under used experimental conditions KNN performs better for bug severity classification. The underperformance of NBM with respect to the KNN algorithm can be understood in terms of their working principle. The KNN algorithm works on the principle of Euclidean distance while the NBM works on the principle of conditional initial probabilities. As the probabilities are calculated during training in NBM, the classifier classifies each data points on the basis of the same initial probabilities while the KNN classifies each data point on the basis of its current distance from its neighbors. Since, this problem requires scores calculated by text mining, each data point may have common values of attributes which is actually not utilized in the NBM.

VI. CONCLUSION AND FUTURE SCOPE

The software bugs that are detected after the deployment of software affect the reliability and quality of software. Bug tracking systems allow users to report these bugs of many open source software. However predicting the severity level of these bug report is emerging issue. Many attempts have been made to address the problem of severity prediction, but no attempt was made for creating dictionary of critical terms of severity indicator. The work presented in this paper is a feature selection and classification approach for categorizing the bug reports into severe and non severe class. Feature selection methods filter out most informative terms from datasets after preprocessing steps. Top 125 terms are selected and used as dictionary terms to train classifier. Bug reports of four components of Eclipse are chosen for this research, four main sub processes performed in experiment are: dataset acquisition, preprocessing, Feature selection and classification. In this research work, after the preprocessing task, matrix of terms and documents are created. Then a feature selection technique is applied over them to get dictionary terms. Two feature selection methods named info – gain and Chi square. Then classification is done by two ML algorithms named as NBM and KNN and on basis of performance matrices their performance is compared

This technique is used on few components of Eclipse for performing severity classification. Therefore in future other components of Eclipse may be used and cross component approach could be applied by creating global dictionary of all components of Eclipse. Also, domain specific projects could be taken into consideration for the study to create global dictionary of domain specific projects. The study could help to make domain specific tool for prediction of severity

REFERENCES

[1] 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.
[2] J. Anvik, L. Hiew, and G. Murphy, “Who should fix thisbug?” in Proc 28th International Conference on SoftwareEngineering. ACM, 2006, pp. 361–370.
[3] Anvik, J. 2007. *Assisting Bug Report Triage through Recommendation*.Ph.D. Dissertation, University of British Columbia
[4] G. Jeong, S. Kim, and T. Zimmermann, “Improving Bug Triage withTossing Graphs,” Proc. 17th ACM SIGSOFT Symp. Foundations ofSoftware Engineering (FSE ’09), Aug. 2009, pp. 111-120.
[5] I. Herraiz, D. German, J. Gonzalez-Barahona, andG. Robles, “Towards a Simplification of the Bug ReportForm in Eclipse,” in 5th International Working Conferenceon Mining Software Repositories, May 2008.
[6] 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.
[7] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, andY.-G. Gu´eh´eneuc, “Is it a bug or an enhancement?:a text-based approach to classify change requests,” inCASCON ’08: Proceedings of the conference of thecenter for advanced studies on collaborative research.ACM, 2008, pp. 304–318
[8] Syed Nadeem Ahsan , Javed Ferzund , Franz Wotawa, “Automatic Software Bug Triage System (BTS) Based on Latent Semantic Indexing and Support Vector Machine”, Proceedings of the 2009 Fourth International Conference on Software Engineering Advances, p.216-221, September 20-25, 2009
[9] Lamkanfi, Ahmed, et al. "Predicting the severity of a reported bug." Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on. IEEE, 2010.
[10] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schröter, and C. Weiss, "What Makes a Good Bug Report?," IEEE Trans. Software Engineering, vol. 36, no.5, Oct. 2010, pp. 618-643.

