www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242

Volume 5 Issue 8 August 2016 Page No. 17710-17714

Proposed Novel Hybrid Approach For Improvement in Bug Severity prediction

Kanchan Rawat[1]***, Dr. Ajay Goel***[2]

[1]Research Scohlar,[2]Professor

[1,2]Department of Computer Science,Baddi University of Emerging Sciences and Technology
Baddi, Solan.H.P-173205, India

[1]rawat.kanchan7@gmail.com

[2] hod.cse@baddiuniv.ac.in

Abstract: *In this research study an approach of creating dictionary of critical terms is proposed 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. The proposed work will help Triager in classifying bugs based on severity and assigning these bugs to relevant developer.*

Keywords**:** Software Bugs, Severity, BTS(Bug tracking System), Feature-Selection, Naive Bayes, K-Nearest Neighbor,

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 aRoyal Air Force Chinook aircraft's engine control computer caused it to crash in the year 1994 and 29 people were killed[3]. So early detection of bugs and their resolution is very critical. Software bug classification helps in bug triaging system. Bug triaging[5] 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

.

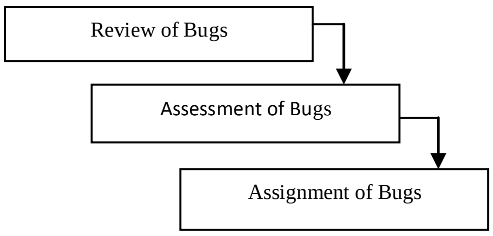

Fig1. Bug Triaging Process

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 [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].

A. Bug:

A Software bug can be classified as error, flaw, failure or fault in any system due to which system behave in an improper manner, may provide results which are not expected or wrong results. Various ways in which a bug can arise are either due to flaws in source code, designing of program or due to operating systems or also can be

DOI: 10.18535/ijecs/v5i8.50

produced by compiler errors. The results of bugs concluded to be hazardous, from various incidents in real world.[9]

B. Types of bugs:

For betterment of software quality it should be ensured that the bugs should be detected and to be taken care in their early stages during software development. Software quality can be affected due to following types of bugs:

i. Arithmetic Bugs: The bugs which are caused by violation of arithmetic rules. Example, divide by zero.
ii. Syntax Bug: The bugs which are caused by the violation of the syntax rules of programming language. Example, using equal to operator instead assignment operator.
iii. Logic Bug: The bugs which are caused by using wrong logic and output is not expected. Example infinite loops.
iv. Resource Bug: The bugs which are caused by in appropriate use of resources. Example, un initialized variable.
v. Multithreading Bug: Example is Deadlock, for multithreading bug.
vi. Interface Bug: Incorrect use of the platform results the interface bug. Examples, incorrect protocol selection.
vii. Performance Bug: Performance bugs degrade the system performance. Example, high computational complexity of an algorithm.

C. Bug tracking System(BTS):

Repositories contain important data regarding the System. In modern software development it is vital to maintain database. these databases can be facilitated to manage the improvement in the software. Bug Tracking Systems are one of the important repositories among all available software repositories which maintain the data regarding the occurrence of bugs, their resolution and their description and many more such attributes regarding the respective bugs occurred in the system. There are various benefits of using Bug Tracking Systems:

i. Helps in improving the quality of the software.
ii. Helps in increasing the customer's satisfaction as it allows them to report it.
iii. User's are not only allowed to report the bugs but they are also provided by the information regarding the fixation of the occurred bug.
iv. It improves the communication between the customers and the developers by reporting bugs and providing the resolution.

II. RELATED WORK

DavorCubanic, Gail C. Murphy [10] made first attempt in 2004 and proposed to apply supervised machine learning algorithms to assists in bug triaging task. This technique can able to detect developer to which the reported bug should be assigned. The presented work is applicable 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. [11] extended the work of DavorCubanic. 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 [12] 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.

GaeulJeong et al.[13] coined a new term called “tossed” (reassignment). Bug tossing signifies the reassignment of bug report to new developer. A tossed graph model has been proposed based on Markov chains and it 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. [14] analyzed the bug reports of Eclipse. 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. [15] in their paper proposed a new automated method called SEVERityISsue assessment (SEVERIS) to assign severity level. It was based on text mining approach and machine learning techniques. The result of case study indicated that SEVERIS was a good method of predicting issue severity levels.

GiulianoAntoniol et al. [16] 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. [17] 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. 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 and co authors [18] 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. [19] 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 proposed 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. METHODOLOGY

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. The whole process of severity classification process proposed in the thesis can be summarized as below:

Step 1: Extraction of Bug Reports of open source software from Bug Repository
Step 2: Pre-processing of Bug reports by using text mining approach
Step 3: Use TF/IDF score for creation of a Term-document matrix(TDM matrix)
Step 4: Use of feature selection methods, information gain and Chi-square method for dimensionality reduction.
Step 5: Creation of critical term dictionary using top-k terms that are obtained after dimensionality reduction and will be fed to the hybrid algorithm proposed in the thesis.
Step 6: Development of Hybrid KNN-NBM algorithm for improved Bug severity prediction.
Step 7: Severity prediction of the bugs reported.
Step 8: Dataset will be used from Bugzilla repository of Mozilla Firefox OS

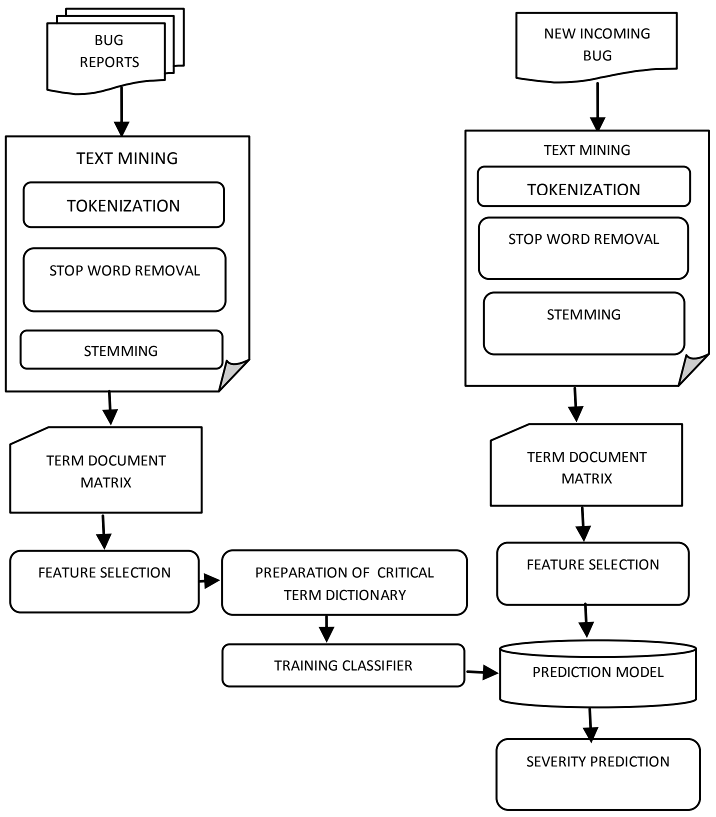

Fig.3 Detailed Methodology

A. Performance Parameters

These are the performance parameters on which our algorithm accuracy, efficiency and complexity would be measured.

i) ***Accuracy****- Accuracy*: is the proportion of the total number of predictions that were correct

- Error rate (misclassification rate)= 1 – AC

$$\textbf{Accuracy} = \frac{\mathrm{TP + TN}}{\mathrm{TP + TN + FP + FN}}$$

- ***Precision****-* Precision or Confidence (as it is called in Data Mining) denotes the proportion of Predicted Positive cases that are correctly Real Positives.

- Precision is defined below:

$$\textbf{Precision} = \text{Confidence } = \frac{\mathrm{tp}}{\mathrm{pp}} = \frac{\mathrm{TP}}{\mathrm{PP}} = \frac{\mathrm{TP}}{\mathrm{TP + FP}}$$

ii) ***Recall***

- Recall or Sensitivity (as it is called in Psychology) is the proportion of Real Positive cases that are correctly Predicted Positive.

$$\textbf{Recall} = \text{Sensitivity} = \text{tpr} = \frac{\mathrm{tp}}{\mathrm{rp}} = \frac{\mathrm{TP}}{\mathrm{RP}} = \frac{\mathrm{TP}}{\mathrm{TP + FN}}$$

B. Comparison Parameter's

These parameters help us in comparing our algorithm with other algorithms and techniques which are used for software quality prediction.

i) ***Confusion Matrix***

- A **table of confusion** (sometimes also called a **confusion matrix**).

- Is a table with two rows and two columns that reports the number of *false positives*, *false negatives*, *true positives*, and *true negatives*.

Table 1: Confusion Matrix

		Predicted Condition	
	Total Population	Predicted Condition (Positive)	Predicted Condition (Negative)
Actual Condition	Actual Condition (Positive)	A: True Positive	B: False Negative
	Actual Condition (Negative)	C: False Positive	D: True Negative

ALGORITHM_NAIVE BAYES MULTINOMIAL

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

Step2. a) For each c ε C

i. Count documents of class c as $N_{c..}$

ii. Calculate prior probability of document of class c.

iii. Count number of occurrence of term t i.e. $T_{ct.}$

b) For each t ε V

i. Calculate conditional probability

$\mathbf{P(t_m/c)} = \frac{\mathbf{T_{ct}+1}}{\mathbf{\sum_{t'\epsilon V} T_{ct'}+B}}$,end for

Step 3. a) For each c ε C, Calculate

$$\mathbf{P\left(\frac{c}{x}\right) = p(c) \prod_{1\leq m\leq n_x} P(t_m/c)}$$

end for

Step 4. Testing document is labeled class that has maximum value of P(c/x)[14][15]

KNN ALGORITHM

Step 1. Compute the distance d from training points (a1, a2, a3….aN) to testing point (t_x) using distance function.

Step 2. Sort the training points according to distance from the training point in ascending order.

Step 3. Top k training points are chosen as nearest neighbor to training points.

Step 4. Most common class level c of k training points is assigned to test point.

IV. PROPOSED METHODOLOGY

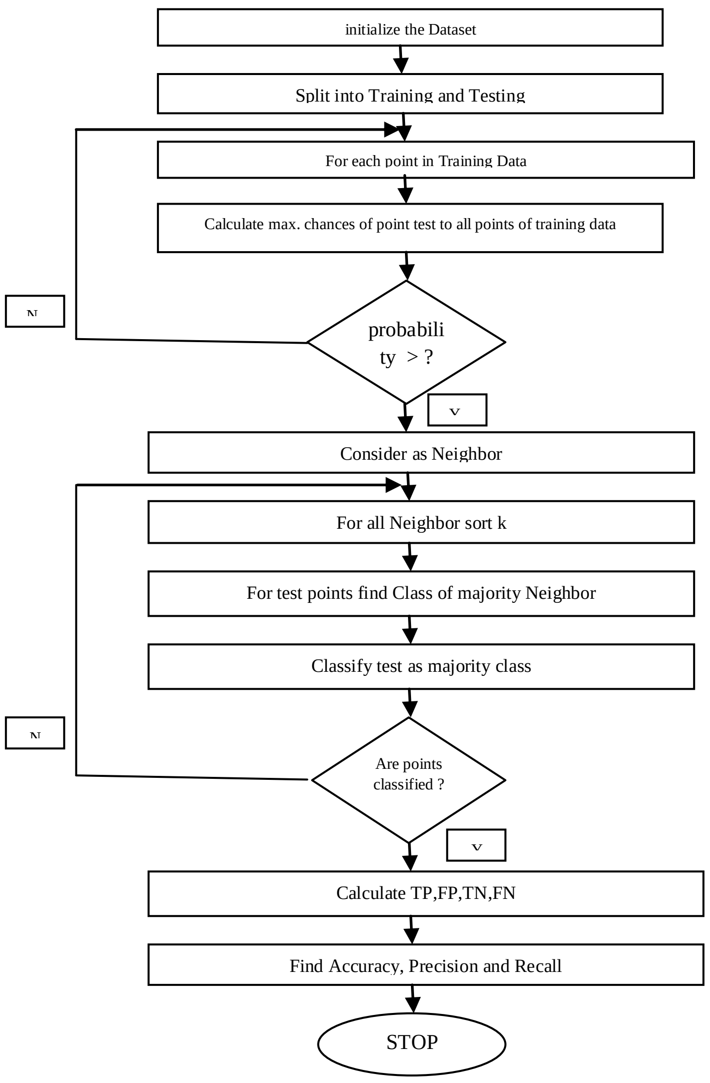
initialize the Dataset
Split into Training and Testing
For each point in Training Data
Calculate max. chances of point test to all points of training data
N
probabili ty > ?
Y
Consider as Neighbor
For all Neighbor sort k
For test points find Class of majority Neighbor
Classify test as majority class
N
Are points classified ?
Y
Calculate TP,FP,TN,FN
Find Accuracy, Precision and Recall
STOP

Fig: PROPOSED METHODOLOGY FLOWCHART

V. 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 proposed 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[16]

The proposed technique is used on bug repository of Mozilla Firefox for performing severity classification. Therefore in future other components of Mozilla may be used and cross component approach could be applied by creating global dictionary of all components of Mozilla. 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]Jonnakuti Ramesh 1 M.Tech II year" An Automated Approach for Software Bug classification " http://ijesat.org/volumes/2013Vol_03_Iss_025/IJSEAT_2013-'03_05_12.

[7]] A. Hotho, A. Nürnberger and G. Paaß, "A Brief Survey of Text Mining," vol. 20, GLDV Journal for Computational Linguistics and Language Technology, 2005, pp. 19-62.

[8] A. E. Hassan, "The Road Ahead for Mining Software Repositories," IEEE Computer society, pp. 48-57, 2008.

[9] S. Diehl, H. C. Gall and A. E. Hassan, "Special issue on mining software repositories," in Empirical Software Engineering An International Journal © Springer Science+Business Media, 2009.

[10] L. Yu, C. Kong, L. Xu, J. Zhao and H. Zhang, "Mining Bug Classifier and Debug Strategy Association Rules for Web-Based Applications," in 08 Proceedings of the 4th international conference on Advanced Data Mining and Applications , 2008.

[11] [Online]. Available: https://bugzilla.mozilla.org.

[12]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.

[13]G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, andY.-G. ´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

[14] 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

[15]Lamkanfi, Ahmed, et al. "Predicting the severity of a reported bug." Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on. IEEE, 2010.

[16]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.

[17] N.Suguna,and Dr. K. Thanushko di"An Improved k-Nearest Neighbor Classification Using Genetic Algorithm" IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 2, July 2010

[18]Yun-leiCai,DuoJi ,Dong-feng Cai"A KNN Research Paper Classification Method Based on Shared Nearest Neighbor" Proceedings of NTCIR-8 Workshop Meeting, June 15–18, 2010, Tokyo, Japan

[19] Li Baoli1, Yu Shiwen, and Lu Qin"An Improved k-Nearest Neighbor Algorithm for Text Categorization" To appear in the Proceedings of the 20th International Conference on Computer Processing of Oriental Languages, Shenyang, China, 2003