International Journal of Research In Science & Engineering e-ISSN: 2394-8299
Volume 3 Issue 1 Jan-Feb 2017 p-ISSN: 2394-8280

IMPROVING BUG TRIAGE WITH DATA REDUCTION USING NB AND KNN CLASSIFIER

Nikita K. Thool[1], Ekta B. Kumari[2], Kiran L. Soni[3], Vaishnavi G. Dhande[4], Prof. Sunny G. Gandhi[5]
Department of IT, D.M.I.E.T.R Wardha

Abstract: Flaw is nothing but an error or bug which produces unexpected and incorrect result. In Companies all software projects are affected by software Flaws (bug).Every day new bugs are generated and developer needs to fix that bug or flaw. Software Company spends lots of money to fix them. Fixing bug is very hard so we will reduce this by using some technique.Every time when bug is generated we need to classify that bug and for that purpose we need classifier. Classifier is the process by which we can classify the bug so that we determine at which class that bug is belong. In this paper we are using two techniques NB (naïve bayes) and KNN (k-nearest neighbor) for classification. NB is based on frequency and KNN is based on word count. After the classification the bug is classified and admin can allot them to the developer to fix. In this paper we also introduce feature selection and instance selection for reducing database. Bug repository is the database which is used to store bug details. In this paper combination of NB and KNN classifier is used which is more efficient and take less time to classify the bug so that admin can assign a proper bug of particular class to the perfect developer AND the bug will fix easily. In the previous paper manual triaging system is used which is not efficient and taking too much time. In this paper we improving flaw triage and also reducing the database by using these two techniques.
Keywords: bug triage, bug data reduction, bug classification technique (NB &KNN).

INTRODUCTION: Most of the companies spend 45% of cost in dealing with the software bugs. This bug waste the time of developer who develop the project. In this paper we improve the flaw triage by using classification technique and also reducing the database .the data which is not useful for fixing the bug we will remove it. This all process is comes under the preprocessing where all the unwanted data is removed and admin get proper data which describe the bug detail. For storing this bug detail we need database this is called bug repository.

Open source software developments incorporate an open bug repository that allows both developers and users to post problems encountered with the software, suggest possible enhancements, and comment upon 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 [12].

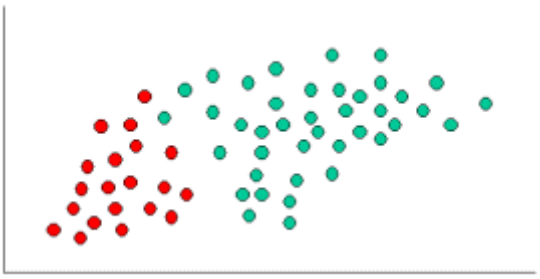
However, this potential advantage also comes with a significant cost. Each bug that is reported must be triaged to determine if it describes a meaningful new problem or enhancement, and if it does, it must be assigned to an appropriate developer for further handling [13].For managing software bugs bug repository or bug fixing plays an important role. Large of software which are open source projects have an open bug repository which allows developers as well as users to submit issues or defects in the software that suggest possible solutions and remark on existing bug reports. The number of regular occurring bugs

for open source large-scale software projects is so much large that makes the triaging process very difficult and challenging .For fixing software bugs most of software companies pays a lot . The large scale and the low quality are main two challenges which are related with bug data that may affect the effective use of bug repositories in software development tasks. Bug is maintained as a bug report in a bug repository that records the reproducing bug in textual form and updates according to the status of bug fixing [1].

This paper introduces Preprocessing and Classification. Preprocessing is the process where unwanted data will remove and we get the useful data for classification and For Classification this two techniques NB (naïve base) and KNN (k nearest neighbor) are use which classify the bug. After applying the classifier, bug will classified.

NB CLASSIFIER

The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.



To demonstrate the concept of Naïve Bayes Classification, consider the example displayed in the illustration above. As indicated, the objects can be classified as either GREEN or RED. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently exiting objects.

Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership GREEN rather than RED. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen.

Thus, we can write:

$$Prior\ probability\ for\ GREEN \propto \frac{Number\ of\ GREEN\ objects}{Total\ number\ of\ objects}$$

$$Prior\ probability\ for\ RED \propto \frac{Number\ of\ RED\ objects}{Total\ number\ of\ objects}$$

KNN CLASSIFIER

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern

recognition already in the beginning of 1970's as a non-parametric technique.

Algorithm

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

It should also be noted that all three distance measures are only valid for continuous variables.
In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and Categorical variables in the dataset.

II. LITERATURE SURVEY

Paper [1]:- Towards Effective Bug Triage with Software Data Reduction Techniques. In this paper a bug repository (a typical software repository, for storing details of bugs), plays an important role in managing software bugs. Software bugs are inevitable and fixing bugs is expensive in software development. Software companies spend over 45 percent of cost in fixing bugs [39]. Large software projects deploy bug repositories (also called bug tracking systems) to support information collection and to assist developers to handle bugs [9], [14]. 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 [64]. A bug repository provides a data platform to support many types of tasks on bugs, e.g., fault prediction [7], [49], bug localization [2], and reopened bug analysis [63]. In this paper, bug reports in a bug repository are called bug data.

Paper [2]:-"Who should fix this bug?"

In this paper they propose open bug repository to which both developers and users can report bugs. The reports that appear in this repository must be triaged to determine if the report is one which requires attention and if it is, which developer will be assigned the responsibility of resolving the report. Large open source development shbare burdened by the rate at which new bug reports appear in the bug repository. In this paper, we present a semi-automated approach intended to ease one part of this process, the assignment of reports to a developer. Our approach applies a machine learning algorithm to the open bug repository to learn the kinds of reports each developer resolves. When a new report arrives, the classifier produced by the machine learning technique suggests a small number of developers suitable to resolve the report. With this approach, we have reached precision levels of 57% and 64% on the Eclipse and Firefox development projects respectively.

Paper [3]:- Finding bugs in web applications using dynamic test generation and explicit-state model checking. In this paper they propose DYNAMIC test generation tools, such as DART [17], Cute[39], and EXE [7], generate tests by executing an application on concrete input values, and then creating additional input values by solving symbolic constraints derived from exercised control-flow paths. To date, such approaches have not been practical in the domain of Web applications, which pose special challenges due to the dynamism of the programming languages, the use of implicit input parameters, their use of persistent state, and their complex patterns of user interaction. This paper extends dynamic test generation to the domain of web applications that dynamically create web (HTML) pages during execution, which are typically presented to the user in a browser.

Paper [4]:- Towards graphical models for text processing. In this paper, they propose the concept of *distance graph representations* of text data. Such representations preserve information about the relative ordering and distance between the words in the graphs, and provide a much richer representation in terms of sentence structure of the underlying data. Recent advances in graph mining and hardware capabilities of modern computers enable us to process more complex representations of text. We will see that such an approach has clear advantages from a qualitative perspective. This approach enables knowledge discovery from text which is not possible with the use of a pure vector-space representation, because itloses much less information about the ordering of the underlying words. Furthermore, this representation does not require the development of new mining and management techniques.

Paper[5]:- Information needs in bug reports: Improving cooperation between In this paper they propose cooperation between developer and user. In open-source projects, bug tracking systems are an important part of how teams (such as the ECLIPSE and MOZILLA teams) interact with their user communities. As a consequence, users can be involved in the bug fixing process: they not only submit the original bug reports but can also participate in discussions of how to fix bugs. Thus they help to make decisions about the future direction of a product. To a large extent, bug tracking systems serve as the medium through which developers and users interact and communicate. However, friction arises when fixing bugs: developers get annoyed and impatient over incomplete bug reports and users are frustrated when their bugs are not immediately fixed [5, 15].

Paper[6]:- Bug Tracking and Reporting System**.**

The paper is wholly dedicated to tracking the bugs that are hither-by arise. The administrator maintains the master details regarding to the bugs id, bugs type, bugs description, bugs severity, bugs status, user details. The administrator too has the authority to update the master details of severity level, status level, etc., modules of the paper. The administrator adds the users and assigns them responsibility of completing the paper. Finally on analyzing the paper assigned to the particular user, the administrator can track the bugs, and it is automatically added to the tables containing the bugs, by order of severity and status. The administrator can know the information in tact the various paper's assigned to various users, their bug tracking status, and their description etc. in the form of reports from time to time. The paper wholly uses the secure way of tracking the system by implementing and incorporating the Server side scripting. The administrator can now add the project modules, project descriptions etc. He too adds the severity level, its status etc.

Paper [7]:- Bug Tracking and Reliability Assessment System (BTRAS).

In this paper they propose a comprehensive classification criteria to review the available tools and propose a new tool named Bug Tracking and Reliability Assessment System (BTRAS) for the bug tracking/reporting and reliability assessment. BTRAS helps in reporting the bug, assigning the bug to the developer for fixing, monitoring the progress of bug fixing by various graphical/charting facility and status updates, providing reliability bug prediction and bug complexity measurements, and distributing fixes to users/developers.

Paper [8]:- Towards Effective Troubleshooting With Data Truncation.

This paper implementing this use instance selection and feature selection for reducing bug of data. And Top-K pruning algorithms for tackling domain specific task. For managing software bugs bug repository or bug fixing plays an important role. Large of software which are open source projects have an open bug repository which allows developers as well as users to submit issues or defects in the software that suggest possible solutions and remark on existing bug reports. The number of regular occurring bugs for open source large-scale software projects is so much large that makes the triaging process very difficult and challenging .For fixing software bugs most of software companies pays a lot . The large scale and the low quality are main two challenges which are related with bug data that may affect the effective use of bug repositories in software development tasks. Bug is maintained as a bug report in a bug repository that records the reproducing bug in textual form and updates According to the status of bug fixing.

Paper [9]:- Stability of Software Defect Prediction in Relation to Levels of Data Imbalance.

The work presented in this paper is a step in that direction. They present a research strategy that aims to explore performance stability of software defect prediction models in relation to levels of data imbalance. As an illustrative example we present an experiment taken to Stability of Software Defect Prediction in Relation to Levels of Data Imbalance our strategy. We observed how learning performance, with and without stepwise feature selection, in case of logistic regression learner, is changing over a range of

imbalances in the context of software defect prediction. The findings are just indicative and are to be explored by exhausting experimenting aligned with proposed strategy.

Paper [10]:- Mining Bug Databases for Unidentified Software Vulnerabilities.

This paper first extends the work of Arnold [1] and clearly demonstrates, empirically, that there is a need for improved identification of which bugs are also vulnerabilities. Then we address the feasibility and elaborate on the difficulties of mining bug databases for discovery of potential hidden impact vulnerabilities. The Linux kernel and My SQL bug databases were chosen for analysis because they have been deployed for many years, have extensive bug and vulnerability databases, and their source code is available for future use in our classification efforts.

III. PROPOSED SYSTEM.

We present the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspects, namely

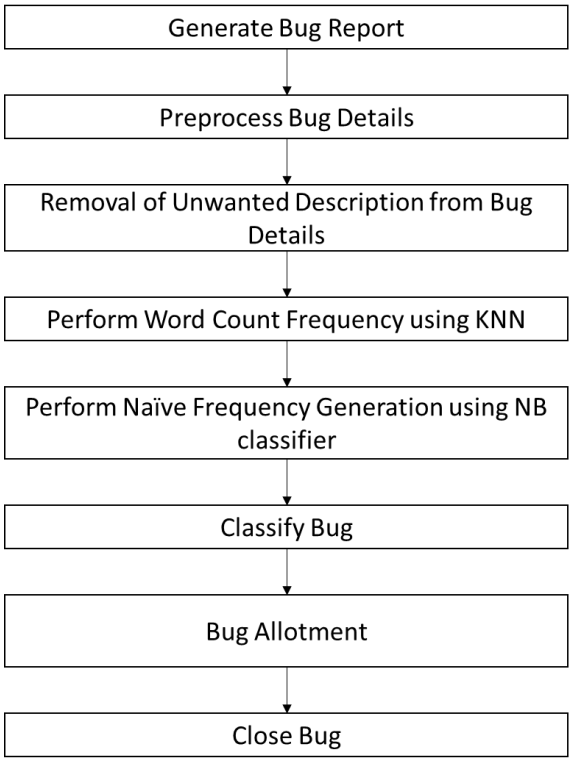
a) To simultaneously reduce the scales of the bug dimension and the word dimension.

b) To improve the accuracy of bug triage.

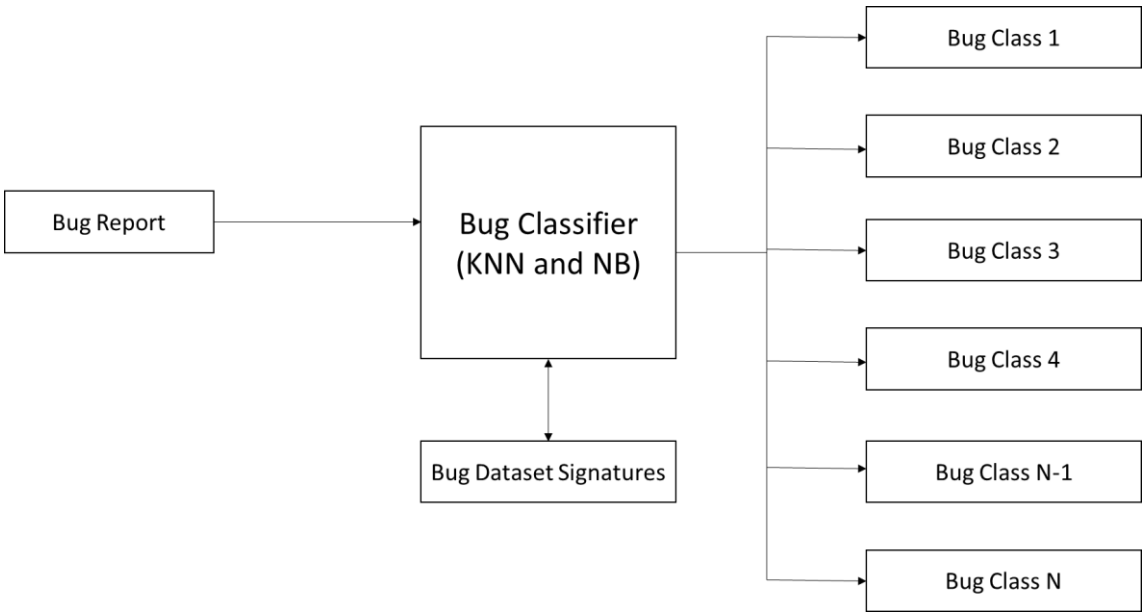
We propose a combination approach to addressing the problem of data reduction. This can be viewed as an application of instance selection and feature selection in bug repositories. We build a combination of NB and KNN classifier to predict the class of the bug. These two techniques are never use is combine form in this paper we using this for increasing efficiency.

Flowchart

e-ISSN: 2394-8299
p-ISSN: 2394-8280



System Architecture



Algorthm:

Algorithm Preprocess (Data D)

Step 1: Read Data into Array

Step 2: Remove All Stop words

$$\Sigma\, i = 0 \mid \phi\, n \neq stop(i)$$

Step 3: Remove Redundancy from Array

$$\Sigma\, i = 0 \mid \phi\, n \neq repeat(i)$$

Step 4: Remove all Special Symbol and digits.

Step 5: Write back

Algorithm Classification (Data D)

Step 1: Read Data into Array

Step 2: Call Preprocess (D)

Step 3: Calculate Word count

$$\Sigma\, i = 0 \mid \phi\, i = i + 1\; if\; a[i] \in final[i]$$

Step 4: Calculate Frequency

Tf = number of occurrences / total words

Step 5: Calculate Normalized TF

NTF = sum of Tf / number of classes

Step 6: Generate Decision Matrix

Step 7: Calculate Final max class value and classify.

CONCLUSION

Bug triage is a costly stride of programming upkeep in both work cost and time cost. In this paper, we join highlight choice with occasion determination to diminish the size of bug information sets and enhance the information quality. To decide the request of applying example determination and highlight choice for another bug information set, we separate traits of every bug information set and prepare a prescient model in view of verifiable information sets. We experimentally examine the information lessening for bug triage in bug storehouses of Find Bugs Database of Bugs. Our work gives a way to deal with utilizing strategies on information handling to shape diminished and brilliant bug information in programming advancement and upkeep. We also provide a system that can classify bugs with

the help of KNN and NB classifier system. Bug Triage with automated classification is the main objective of proposed system

REFERENCES

[1] Jifeng Xuan, He Jiang, Member, IEEE, Yan Hu, ZhileiRen, WeiqinZou, ZhongxuanLuo, and Xindong Wu, Fellow, IEEE" Towards Effective Bug Triage with Software Data Reduction Techniques" VOL. 27, NO. 1, JANUARY 2015.

[2] 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 .

[3] 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.

[4] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

[5] 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.

[6] A.S.SyedFiaz, N.Devi, S.Aarthi"Bug Tracking and Reporting System**"**International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-1, March 2013.

[7] V.B. Singh1, Krishna Kumar Chaturvedi2 "Bug Tracking and Reliability Assessment System (BTRAS)" International Journal of Software Engineering and Its Applications Vol. 5 No. 4, October, 2011.

[8] Karishma Musale1, GorakshanathGagare **2"**Towards Effective Troubleshooting With Data Truncation" *International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 11, November 2015.*

[9] TIHANA GALINAC GRBAC AND GORAN MAUˇ SA, **"**Stability of Software Defect Prediction in Relation to Levels of Data Imbalance" University of RijekaBOJANA DALBELO–BASˇ ICˊ , University of Zagreb.

[10] DumiduWijayasekara Milos Manic Jason L. Wright Miles McQueen "Mining Bug Databases for Unidentified Software Vulnerabilities".

[11] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," ACM Trans. Soft. Eng. Method, vol. 20, no. 3, article 10, Aug. 2011.

[12] E. S. Raymond. The cathedral and the bazaar. First Monday, 3(3), 1998.

[13] C. R. Reis and R. P. de Mattos Fortes. An overview of the software engineering process and tools in the
Mozilla project. In Proceedings of the Open Source Software Development Workshop, pages 155–175, 2002.

[14] Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/

[15] K. Balog, L. Azzopardi, and M. de Rijke, "Formal models for expert finding in enterprise corpora," in Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, Aug. 2006, pp. 43–50.

[16] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Trans. Knowl. Data Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.
[17] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," Data Mining Knowl. Discovery, vol. 6, no. 2, pp. 153–172, Apr. 2002.