Improving Flaw Triage with Data Reduction using NB and KNN Classifier

[1]Nikita R. Hiwse, [2]Prof. Roshani Talmale, [3]Prof. Jayant Rohankar

[1]Wireless Communication And Computing,[2]Computer Science & Engineering, [3]Information Technology

Tulsiramji Gaikwad-Patil College of Engineering &Technology, Nagpur

Email(***Mandatory***): [1]hiwsenikita04@gmail.com, [2]hod.cse@tgpcet.com[3] jsrohankar@gmail.com

Contact Number(***Mandatory***):: 7709191219[1],9011009866[2] +91-8554982302[2]

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. SoftwareCompanyspends 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).*

*****

I. INTRODUCTION

Mostofthe 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 significantcost. Each bug that is reported must be triagedto 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].

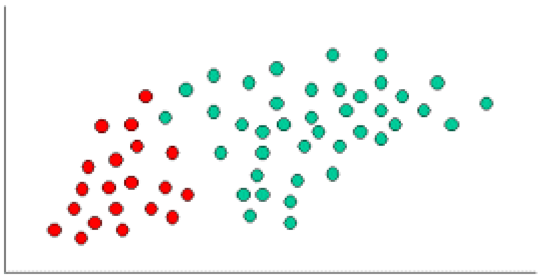
For managing software bugs bug repository or bug fixing plays an important role. Large of software which are opensource projects have an open bug repository which allows developers as well as users to submit issues or defects inthe software that suggest possible solutions and remark on existing bug reports. The number of regular occurring bugs for open source large-scale software projects is so much large that makes the triaging process very difficultand challenging .For fixing software bugs most of software companies pays a lot . The large scale and thelow quality are main two challenges which are related with bug data that may affect the effective use of bugrepositories in software development tasks. Bug is maintained as a bug report in a bug repository that recordsthe 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 system depends on the supposed Bayesian hypothesis and is especially suited when the dimensionality of the inputs is high. In spite of its straightforwardness, Naive Bayes can frequently outflank more refined grouping techniques.

To exhibit the idea of Naïve Bayes Classification, consider the case showed in the outline above. As showed, the articles can be named either GREEN or RED. Our errand is to group new cases as they arrive, i.e., choose to which class name they have a place, taking into account the at present leaving objects.
Since there are twice the same number of GREEN articles as RED, it is sensible to trust that another case (which hasn't been watched yet) is twice as prone to have participation GREEN as opposed to RED. In the Bayesian examination, this conviction is known as the earlier likelihood. Earlier probabilities depend on past experience, for this situation the rate of GREEN and RED items, and frequently used to foresee results before they really happen.



Thus, we can write:

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

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

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

II. LITERATURE SURVEY

Paper [1]:- Towards Effective Bug Triage with Software Data Reduction Techniques. In this paper a bug storehouse (a commonplace programming vault, for putting away subtle elements of bugs), assumes a critical part in overseeing programming bugs. Programming bugs are unavoidable and altering bugs is costly in programming advancement. Programming organizations spend more than 45 percent of expense in settling bugs [39]. Substantial programming ventures send bug storehouses (likewise called bug following frameworks) to bolster data accumulation and to help designers to handle bugs [9], [14]. In a bug vault, a bug is kept up as a bug report, which records the literary depiction of replicating the bug and upgrades as indicated by the status of bug altering [64]. A bug vault gives an information stage to bolster numerous sorts of errands on bugs, e.g., issue forecast [7], [49], bug confinement [2], and revived bug investigation [63]. In this paper, bug reports in a bug vault are called bug information.

Paper [2]:- "Who ought to settle this bug?"
In this paper they propose open bug archive to which both engineers and clients can report bugs. The reports that show up in this store must be triaged to figure out whether the report is one which requires consideration and in the event that it is, which designer will be doled out the obligation of determining the report. Huge open source advancement shbare loaded by the rate at which new bug reports show up in the bug store. In this paper, we display a semi-computerized approach planned to straightforwardness one a player in this procedure, the task of reports to an engineer. Our methodology applies a machine learning calculation to the open bug store to take in the sorts of reports every engineer determines. At the point when another report arrives, the classifier created by the machine learning strategy recommends a little number of designers appropriate to determine the report. With this methodology, we have achieved accuracy levels of 57% and 64% on the Eclipse and Firefox improvement extends individually.

Paper [3]:- Finding bugs in web applications utilizing dynamic test era and express state model checking. In this paper they propose DYNAMIC test era apparatuses, for example, DART [17], Cute[39], and EXE [7], produce tests by executing an application on solid info qualities, and after that making extra information values by fathoming typical requirements got from practiced control-stream ways. To date, such methodologies have not been pragmatic in the space of Web applications, which posture extraordinary difficulties because of the dynamism of the programming dialects, the utilization of understood information parameters, their utilization of tireless state, and their unpredictable examples of client communication. This paper stretches out element test era to the space of web applications that progressively make web (HTML) pages amid execution, which are ordinarily displayed to the client in a program.

Paper [4]:- Towards graphical models for content preparing. In this paper, they propose the idea of separation chart representations of content information. Such representations save data about the relative requesting and separation between the words in the charts, and give a much wealthier representation as far as sentence structure of the basic information. Late advances in chart mining and equipment abilities of present day PCs empower us to process more mind boggling representations of content. We will see that such a methodology has clear points of interest from a subjective viewpoint. This methodology empowers learning disclosure from content which is impractical with the utilization of an immaculate vector-space representation, in light of the fact that itloses a great deal less data about the requesting of the hidden words. Besides, this representation does not require the advancement of new mining and administration systems.

Paper[5]:- Information needs in bug reports: Improving collaboration between In this paper they propose participation amongst designer and client. In open-source ventures, bug following frameworks are a critical piece of how groups, (for example, the ECLIPSE and MOZILLA groups) associate with their client groups. As an outcome, clients can be included in the bug altering process: they present the first bug reports as well as take part in exchanges of how to settle bugs. Consequently they settle on choices about the future course of an item. To a substantial degree, bug following frameworks serve as the medium through which designers and clients associate and convey. Be that as it may, grinding emerges when altering bugs: engineers get irritated and fretful over inadequate bug reports and clients are disappointed when their bugs are not instantly settled [5, 15].

Paper[6]:- Bug Tracking and Reporting System.
The paper is entirely committed to following the bugs that are here by emerge. The manager keeps up the expert insights with respect to the bugs id, bugs sort, bugs portrayal, bugs seriousness, bugs status, client points of interest. The head too has the power to redesign the expert points of interest of seriousness level, status level, and so forth., modules of the paper. The manager includes the clients and doles out them obligation of finishing the paper. At last on investigating the paper doled out to the specific client, the chairman can track the bugs, and it is naturally added to the tables containing the bugs, by request of seriousness and status. The overseer can know the data in judgment the different paper's relegated to different clients, their bug following status, and their depiction and so forth as reports now and again. The paper entirely utilizes the protected method for following the framework by actualizing and joining the Server side scripting. The manager can now include the venture modules, venture depictions and so forth. He too includes the seriousness level, its status and so on.

Paper [7]:- Bug Tracking and Reliability Assessment System (BTRAS).
In this paper they propose an extensive characterization criteria to audit the accessible instruments and propose another apparatus named Bug Tracking and Reliability Assessment System (BTRAS) for the bug following/reporting and unwavering quality evaluation. BTRAS helps in reporting the bug, doling out the bug to the engineer for settling, observing the advancement of bug altering by different graphical/outlining office and announcements, giving dependability bug expectation and bug unpredictability estimations, and conveying fixes to clients/designers.

Paper [8]:- Towards Effective Troubleshooting With Data Truncation.
This paper actualizing this utilization example determination and highlight choice for diminishing bug of information. What's more, Top-K pruning calculations for handling space particular assignment. For overseeing programming bugs bug vault or bug altering assumes an imperative part. Expansive of programming which are open source ventures have an open bug vault which permits designers and also clients to submit issues or absconds in the product that recommend conceivable arrangements and comment on existing bug reports. The quantity of consistent happening bugs for open source extensive scale programming activities is so much huge that makes the triaging procedure exceptionally troublesome and testing .For altering programming bugs the majority of programming organizations pays a great deal . The expansive scale and the low quality are principle two difficulties which are connected with bug information that may influence the viable utilization of bug archives in programming improvement errands. Bug is kept up as a bug report in a bug storehouse that records the duplicating bug in printed shape and upgrades As indicated by the status of bug settling.

Paper [9]:- Stability of Software Defect Prediction in Relation to Levels of Data Imbalance.
The work displayed in this paper is a stage in that heading. They exhibit an exploration system that means to investigate execution security of programming deformity forecast models in connection to levels of information lopsidedness. As an illustrative case we display a trial taken to Stability of Software Defect Prediction in Relation to Levels of Data Imbalance our methodology. We watched how learning execution, with and without stepwise component determination, if there should arise an occurrence of logistic relapse learner, is changing over a scope of uneven characters with regards to programming imperfection expectation. The discoveries are simply characteristic and are to be investigated by debilitating testing adjusted to proposed system.

Paper [10]:- Mining Bug Databases for Unidentified Software Vulnerabilities.
This paper first develops the work of Arnold [1] and unmistakably illustrates, exactly, that there is a requirement for enhanced recognizable proof of which bugs are likewise vulnerabilities. At that point we address the achievability and expound on the challenges of digging bug databases for disclosure of potential shrouded sway vulnerabilities. The Linux piece and My SQL bug databases were decided for

investigation since they have been sent for a long time, have broad bug and weakness databases, and their source code is accessible for future use in our order endeavors.

III. PROPOSED SYSTEM.

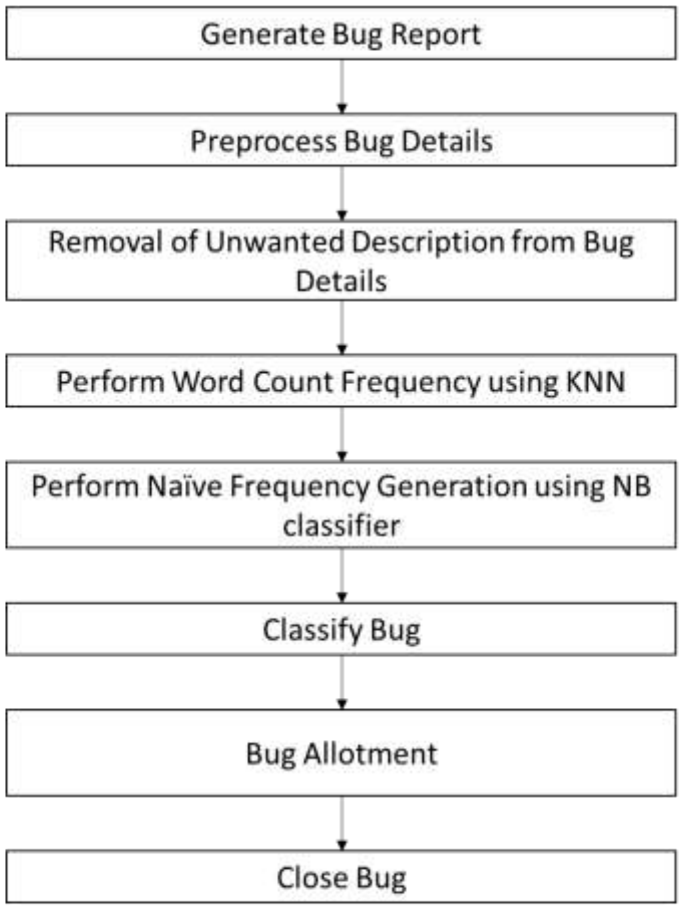
We present the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspects, namely

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

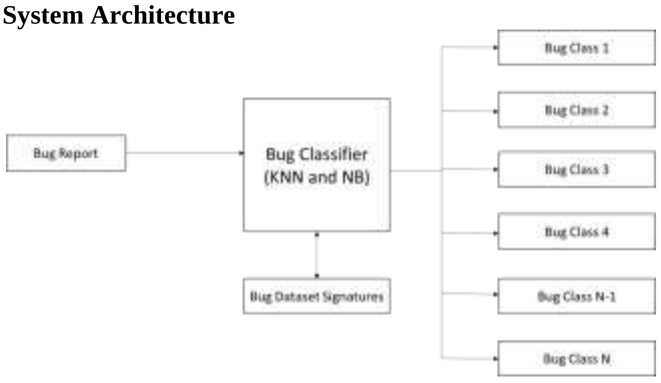
b) To improve the accuracy of bug triage.

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


Generate Bug Report
Preprocess Bug Details
Removal of Unwanted Description from Bug Details
Perform Word Count Frequency using KNN
Perform Naïve Frequency Generation using NB classifier
Classify Bug
Bug Allotment
Close Bug

System Architecture



Algorithm:

```
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 Wordcount
        $\Sigma i = 0 \mid \phi\, i = i + 1\ if\ a[i]\ \in final[i]$
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 FindBugs 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 IC´ , 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.