Effective bug triage with Prim's algorithm for Feature Selection

*Snehal Chopade
Department of Computer Engineering,
G.H. Raisoni College of Engineering and Management,Wagholi, Pune 412207
snehalchopade8@gmail.com
**Prof. Pournima More
Department of Computer Engineering,
G.H. Raisoni College of Engineering and Management,Wagholi, Pune 412207
pournima.more18@gmail.com

***Abstract* — For constructing any software application or item it is essential to identify the bug in the item while building up the product. At every stage of testing the bug report is created, more of the time is wasted for settling the bug. In settling the bug, software enterprises waste 45 percent of cost. One of the basic systems for settling the bug is bug triage. It is a process for settling the bugs whose fundamental object is to properly allocate a designer to a novel bug for further taking handling. If the assigned developer is busy or not available then this system assigns a new domain specific developer. Initially manual work is accomplished for each time creating the bug report. After that content analysis strategies are functional to conduct normal bug triage. The current framework resists the problem of data reduction in the settling of bugs naturally. Furthermore there is a need of methods which decreases the range likewise enhances the excellence of bug data. For feature collection which is not given precise outcome traditional framework utilized CHI technique. In this way this framework proposed the technique for feature selection by utilizing the Prim's strategy. By joining the instance selection and the feature selection calculations to simultaneously diminish the data scale likewise upgrade precision of the bug reports in the bug triage. By utilizing Prim's strategy, noisy words are removed from the dataset set. From the experimental result it is displayed that accuracy of the proposed system is greater than the accuracy of the existing system.**

Keywords— Bug Triage, feature selection, instance selection, Prim's algorithm.

I. INTRODUCTION

A bug repository assumes a significance part in managing the software bugs. Many open source software projects have an open bug repository that permits both developers and clients to submit imperfections or issues in the product, propose conceivable improvements, and remark on existing bug reports. For open source extensive scale software projects, the number of daily bugs is so substantial which makes the triaging procedure extremely troublesome and challenging [2]. Software organizations spend more than 45 percent of cost in fixing bugs .There are two difficulties identified with bug information that may influence the effective utilization of bug repositories in software development assignments, in particular the expansive scale and the low quality. In a bug repository, a bug is kept up as a bug report, which records the textual depiction of reproducing the bug and updated as per the status of bug fixing [1].

Main aim of bug triage is to assign a developer for bug settling. Once a developer is allotted to another bug report developer will settle the bug or attempt to amend it. Developer will give the status identified with bug whether it is redressed or not [1].

Instance Selection: This strategy connected with information mining tasks for example, characterization and clustering

- It's a nontrivial procedure of recognizing substantial, novel, potentially valuable, and eventually justifiable examples in data. Selecting a subset of information to accomplish the original purpose of the data mining application as though the entire information is utilized.
- The perfect result of instance selection is model independent.

This scheme is proposed to develop an effective model for doing data reduction on bug data set for reducing the range of the information also improve the excellence of the data by falling the time and cost of bug reducing techniques. Proposed improved feature selection method for tending the issues for reduction of information. The main output of this system is:

- For removing the noisy words from the dataset feature collection is used.
- Instance collection can remove uninformative bug reports.

- The accuracy of the bug triage is improved by eliminating the redundant words;
- Instance collection can recover the accuracy loss.

In this paper we study about the related work done, in section II, the proposed approach modules description, mathematical modeling, algorithm and experimental setup in section III .and at final we provide a conclusion in section IV.

II. LITERATURE REVIEW

In this section discuss the literature review in detail about the recommendation system for online social network.

Jifeng Xuan et. al. [1] tackle the issue of information diminishment for bug triage, i.e., how to decrease the scale and enhance the nature of bug information. They merge instance selection with feature selection to simultaneously diminish information scale on the bug measurement and the word measurement. To decide the request of applying instance selection and feature selection, they remove attributes from historical bug information sets and build a predictive model for bug information set.

D. Cubranic and G. C. Murphy [2] propose to apply machine learning systems to help with bug triage by utilizing content categorization to anticipate the designer that should work at the bug on the basis of the bug's depiction.

W. Zou et. al. [3] proposed the training set decrease with both feature selection and instance selection strategies for bug triage. They merge feature selection with instance selection to enhance the exactness of bug triage. The feature selection protocol χ 2 - test, instance selection protocol Iterative Case Filter, and their combinations are examined in this paper.

Gaeul Jeong et. al. [4] presents a graph model on the basis of Markov chains, which catches bug tossing history. This model has a few desirable qualities. To begin with, it uncovers designer systems which can be utilized to find team structures and to discover reasonable specialists for a new task. Second, it assists to better assign developers to bug reports.

J. Xuan et. al. [5] propose a semi-supervised content categorization approach for bug triage to maintain a strategic distance from the inadequacy of named bug reports in existing supervised approaches. This new approach joins Bayes classifier and exception maximization to exploit both labeled and unlabeled bug reports. This approach trains a classifier with a fraction of labeled bug reports. At that point the approach iteratively marks various unlabeled bug reports and prepares another classifier with names of all the bug reports.

V. Cerveron and F. J. Ferri [6] introduces novel approach to deal with the determination of models for the nearest neighbor rule which aims at getting an optimal or near to optimal solution. The issue is expressed as a constrained optimization issue utilizing the idea of consistency. In this specific situation, the proposed technique utilizes tabu search as a part of all conceivable subset.

S. Breu et. al. [7] has quantitatively and subjectively examined the questions asked in a sample from 600 bug reports from the mozilla and eclipse projects. We arranged the questions and examined reaction rates and times by classification and project.

J.W. Park et. al. [8] proposes a costaware triage protocol, COSTRIAGE. COSTRIAGE models "developer profiles" to show developers assessed costs for settling distinctive types of bugs. Types of bugs are separated by applying Latent Dirichlet Allocation (LDA) to bug report corpora. This tackles the inadequacy issue and improves the proposal quality of CBCF

III. PROPOSED APPROACH

A. Proposed System Overview

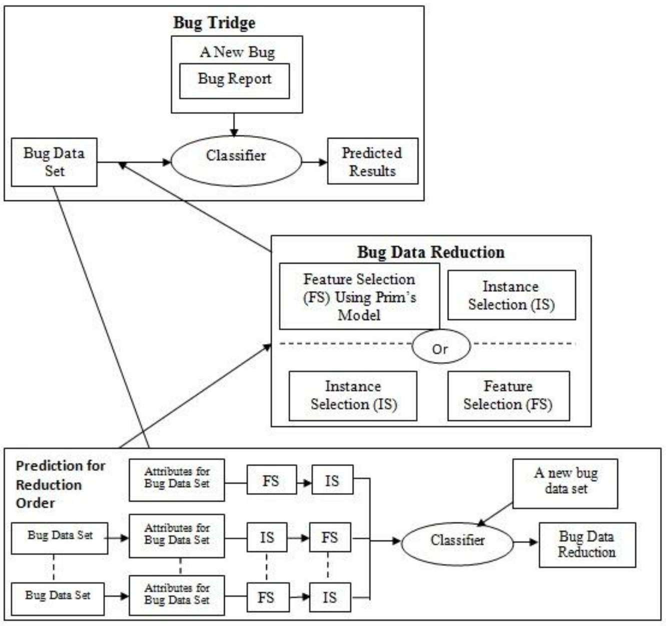

Figure 1.Proposed System Architecture

Techniques used to implement this system:

Bug Triage block shows the architecture of work done by the existing scheme on bug triage. In this system initially a phase of data reduction in added after that classifier is trained a with a bug data set. Big data reduction block unite the method of instance collection and feature collection to diminish the range of bug information. In the process of bug data reduction, the issues are how to conclude the order of two reduction methods. Prediction for reduction order on the basis of the attributes of historical bug data sets, system introduced a binary categorization method to forecast decrease orders. Also, propose the improved feature collection technique by using prim's model for tackling the issues of data reduction.

- Uploading Input File:

New bug dataset is taken as input initially. After uploading the input file.

- Attribute Selection

Attributes are selected from the dataset. The dataset contain features and attributes.

- Classification

The process of classification is performed for obtain the instance selection and feature selection. Features are selected by using the prim's method. From this process the bug data reduction output is obtained.

- Predicted Result

The result obtained from the last step and again the classification process is done. And finally the predicted result is obtained.

B. Algorithm

Algorithm 1: Proposed Algorithm

Step1: Read data set.

Step2: Prediction of Reduction Order from Historical Dataset by using Naive Bayes Classifier.

Step3: If class value for Reduction Order is equal to 1 then Reduction Order will be FS to IS and if class value is 0 then Reduction Order will be IS to FS.

Step4: Feature Selection using Prims Algorithm to reduce number of attributes in the dataset.

Step5: Instance Selection using ICF to reduce number of instances in the dataset.

Step6: Classification for testing data using Naive Bayes Classifier on reduced dataset after Feature Selection and Instance Selection. Developer will be assigned to testing data based on reduce dataset.

Step7: If the assigned developer is busy then assign new developer to testing data.

Algorithm 2: Prim's Algorithm

The algorithm may informally be described as performing the following steps:

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.
2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.
3. Repeat step 2 (until all vertices are in the tree).

In more detail, it may be implemented following the Pseudo code below.

```
1. Associate with each vertex $v$ of the graph a number $C[v]$ (the cheapest cost of a connection to $v$)
   and an edge $E[v]$ (the edge providing that cheapest connection). To initialize these values, set all
   values of $C[v]$ to $+\infty$ (or to any number larger than the maximum edge weight) and set each $E[v]$
   to a special flag value indicating that there is no edge connecting $v$ to earlier vertices.
2. Initialize an empty forest $F$ and a set $Q$ of vertices that have not yet been included in $F$
   (initially, all vertices).
3. Repeat the following steps until $Q$ is empty:
      a. Find and remove a vertex $v$ from $Q$ having the minimum possible value of $C[v]$
      b. Add $v$ to $F$ and, if $E[v]$ is not the special flag value, also add $E[v]$ to $F$
      c. Loop over the edges $vw$ connecting $v$ to other vertices $w$. For each such edge, if $w$ still
         belongs to $Q$ and $vw$ has smaller weight than $C[w]$, perform the following steps:
            i. Set $C[w]$ to the cost of edge $vw$
           ii. Set $E[w]$ to point to edge $vw$.
4. Return $F$
```

IV. RESULTS AND DISCUSSION

A. Experimental Setup

The system is built using Java framework on Windows platform. The Net beans IDE are used as a development tool. The system doesn't require any specific hardware to run; any standard machine is capable of running the application.

B. Dataset

In this system, bug data set of Mozilla and eclipse are used.

Table 1: Data Sets in Mozilla and Eclipse

	Name	DS-E1	DS-E2	DS-E3	DS-E4	DS-E5
Eclipse	Range of Bug IDs	200001 - 220000	220001 - 240000	240001 - 260000	260001 - 280000	280001 - 300000
	# Bug reports	11,313	11,788	11,495	11,401	10,404
	# Words	38,650	39,495	38,743	38,772	39,333
	# Developers	266	266	286	260	256
Mozilla	Name	DS-M1	DS-M2	DS-M3	DS-M4	DS-M5
	Range of Bug IDs	400001 - 440000	440001 - 480000	480001 - 520000	520001 - 560000	560001 - 600000
	# Bug reports	14,659	14,746	16,479	15,483	17,501
	# Words	39,749	39,113	39,610	40,148	41,577
	# Developers	202	211	239	242	273

C. Result

In this section, the experimental result of the proposed system is discussed.

In table 2 shows the accuracy of proposed and existing system. CHI square and Prim's algorithm are used for feature selection of the dataset. The result obtained or the feature selected from the CHI square method is less accurate than the result obtained from the Prim's algorithm. For instance selection ICF algorithm is used.

The accuracy is calculated as:

$$Accuracy = \frac{\#\, correctly\ predicted\ orders}{\#\, all\ datasets}$$

Table 2: Accuracy of FS on DS.E1

List Size	FS	
	CHI Square	Prim's
1	32.71	36.88
2	44.97	50.63
3	51.73	59.23
4	56.58	61.47
5	60.40	67.32

Following figure 2 shows the comparison of proposed system and existing system on the basis of their accuracy for selection of features. From the graph it shows that accuracy of the proposed system is more than the accuracy of the existing system.

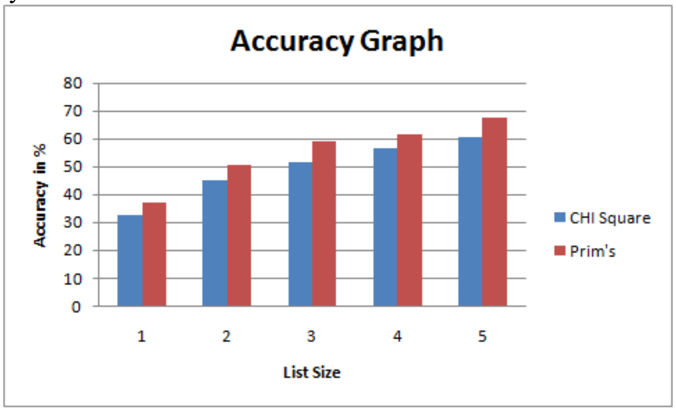

Fig 2. Accuracy Graph

List Size- it is recommended list. Given bug dataset, we sort the developers by number of their fixed bugs in descending order. That is we sort classes by number of instances in classes.

V.CONCLUSION AND FUTURE SCOPE

Bug triage is an expensive step of keep up the labor cost and time cost of the product. In this scheme, join feature selection with instance selection to decreases the range of bug datasets and in addition improve the quality data. This paper introduce the Prim's Method which is exploited for feature selection rather than CHI feature selection technique which divides attributes of each bug dataset and prepare a model on the basis of historical datasets. System offers an approach to deal with utilizing procedures on data processing to form reduced and high-utility bug data in software advancement and support. The estimation result display that the accuracy of the proposed plan is greater than the accuracy of the existing plan.

REFERENCES

1. Jifeng Xuan, He Jiang, Member, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, and Xindong Wu, "Towards Effective Bug Triage with Software Data Reduction Techniques", IEEE Transcation on knowledge and data engineering, vol.27, No 1, january 2015.
2. D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.
3. W. Zou, Y. Hu, J. Xuan, and H. Jiang, "Towards training set reduction for bug triage," in Proc. 35th Annu. IEEE Int. Comput. Soft. Appl. Conf., Jul. 2011, pp. 576–581.
4. Gaeul Jeong, Sunghun Kim, Thomas Zimmermann, ``Improving Bug Triage with Bug Tossing Graphs", ESEC-FSE'09, August 23–28, 2009
5. J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Automatic bug triage using semi-supervised text classification," in Proc. 22nd Int. Conf. Softw. Eng. Knowl. Eng., Jul. 2010, pp. 209–214.
6. V. Cerveron and F. J. Ferri, "Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule," IEEE Trans. Syst., Man, Cybern., Part B, Cybern., vol. 31, no. 3, pp. 408-413, Jun. 2001.
7. 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.
8. J.W. Park, M.W. Lee, J. Kim, S.W. Hwang, and S. Kim, "Costriage: A cost-aware triage algorithm for bug reporting systems," in Proc. 25th Conf. Artif. Intell., Aug. 2011, pp. 139-144.