Research Article Impact Factor: 4.226 ISSN: 2319-507X
S. A. Ladhake, IJPRET, 2016; Volume 4 (9): 1045-1058 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

EFFECTIVE BUG TRACKING AND REPORTING SYSTEM USING NAIVE BAYES APPROACH

DR. S. A. LADHAKE[1], MISS. PALLAVI A. KHARODE[2]

1. Principal, SIPNA COET AMRAVATI, Maharashtra, India
2. Student, M.E Computer Science and Engineering, SIPNA COET AMRAVATI, Maharashtra, India.

Accepted Date: 15/03/2016; Published Date: 01/05/2016

Abstract*:* In software industries, individuals at different levels from customer to an engineer apply diverse mechanisms to detect to which class a particular bug should be allocated. Sometimes while a simple search in Internet might help, in many other cases a lot of effort is spent in analysing the bug report to classify the bug. So there is a great need of a structured mining algorithm - where given a crash log, the existing bug database could be mined to find out the class to which the bug should be allocated. This would involve Mining patterns and applying different classification algorithms. This paper focuses on the feature extraction, noise reduction in data and classification of network bugs using probabilistic Naïve Bayes approach. Different event models like Bernoulli and Multinomial are applied on the extracted features. When new unseen bugs are given as input to the algorithms, the performance comparison of different algorithms is done on the basis of accuracy and recall parameters.

Keywords: Classification, Multinomial Model, Bayesian, Network Bugs.

Corresponding Author: DR. S. A. LADHAKE

Access Online On:

www.ijpret.com

How to Cite This Article:

S. A. Ladhake, IJPRET, 2016; Volume 4 (9): 1045-1058

PAPER-QR CODE

INTRODUCTION

With the advancement in software technology, as number of software products are increasing, maintenance is becoming a challenging task. Maintenance activities account for over two third of the life cycle cost of a software system [4]. Hence a lot of time and efforts are required for maintenance phase of software development lifecycle. Essential activities involved in maintenance are bug reporting and fixing [1]. Many developers put significant amount of effort for finding and debugging software bugs.

In addition, a significant amount of submitted bug reports are duplicates that describe already-reported bugs [2]. Sometimes while a simple search in Internet might help, in many cases a lot of effort is spent in analysing the bug reports to classify the bug.

In practice, considerable number of the duplicate bugs is reported daily; manually labeling these bugs is highly time consuming task. To address above mentioned issue, several techniques have been proposed using various similarity based metrics. The similarity measures are used to detect candidate duplicate bug reports or identical bugs and to classify bugs [3]. The same bug can be reported in two different ways and hence extracting features and classifying the bug reports become very complicated.

Different discriminative model based approaches are also used for classifying [2]. But all these approaches are generalized, which can be applied to any type of bug database and gives an accuracy of around 30-35% [2]. The accuracy can still be improved if bug semantics are taken into consideration.

The bug classification is highly dependent on the type and characteristic of the bugs. For example, network bugs have different semantics as compared to bugs related to IDE's (ECLIPSE, Netbeans etc) or bugs related to browsers (Mozilla, Chrome). Feature Extraction based on severity, priority and other general information is independent of the types of bugs. We think that if the feature extraction is done on the basis of the bug specific characteristic then efforts of the developer for bug fixing can be minimized.

In this paper, we analyzed the network bugs and depending on the static analysis of the bug reports, the feature extraction and selection has been performed. Feature Extraction from bug report is performed according to networking protocols, operating system related defects, product related bugs and different networking protocols to which they belong such as Border Gateway Protocol (BGP), Internet Protocol (IPv4 and IPv6), and Transmission Control Protocol

(TCP) etc. Different bug specific features are ranked according to Information Gain criteria. Two different event models: Bernoulli and Multinomial Model can be considered for classification.

2. LITERATURE REVIEW

Davor Cubranic and Gail C.Murphy have proposed an approach for automatic bug triage using text categorization [5]. They proposed a prototype for bug assignment to developer using supervised Bayesian learning. The evaluation shows that their prototype can correctly predict 30% of the report assignments to developers. The prototype used the word frequency as input to the classifier. The words were extracted using Natural Language Processing Techniques. The words can be considered as unigram features obtained irrespective of the type of bugs. They analyzed their technique on open source eclipse bug database. Nicholas Jalbert and Westley Weimer have proposed a system that automatically classifies duplicate bug reports as they arrive to save developer time [6]. Their system used surface features, textual semantics, and graph clustering to predict duplicate status. Using a dataset of 29,000 bug reports from the Mozilla project, they performed experiments that include a simulation of a real-time bug reporting environment. Their system was able to reduce development cost by filtering out 8% of duplicate bug reports while allowing at least one report for each real defect to reach developers. Deqing Wang, Mengxiang Lin, Hui Zhang, Hongping Hu have implemented a tool Rebug-Detector, to detect related bugs using bug information and code features[7]. They extracted features related to bugs and used relationship between different methods that is overloaded or overridden methods. They evaluated Rebug-Detector on an open source project: Apache Lucene-Java. The results show that bug features and code features extracted by their tool are useful to find real bugs in existing projects. The classification of the bugs into different buckets can be done using data mining and machine learning concepts. The bugs will be classified into different buckets according to selected features. The approach of bucketing was used by Microsoft for their Windows Error Reporting System [8]. Windows Error Reporting (WER) is a distributed system that automates the processing of error reports coming from an installed base of a billion machines. It collects error data automatically and classifies errors into buckets, which are used to prioritize developer effort and report fixes to users. For Bucketing two types of heuristics were applied: Expanding heuristics increase the number of buckets with the goal that no two bugs are assigned to the same bucket, Condensing heuristics decrease the number of buckets with the goal that no two buckets contain error reports from the same bug. The two heuristics are complementary to each other and ensure the correct and efficient bucketing. Karl-Michael Schneider in the paper[10] used Naive Bayes Method for Spam Classification. Kian Ming Adam Chai, Hwee Tou Ng and Hai Leong Chieus in their paper [9], explores the use of Bayesian probability approach for text classification. They had used ltc normalization and compared two different types of Bayesian classifiers that is Bayesian Online perception and Bayesian Gaussian Process. They showed through experiments that Bayesian is good approach for text classification.

3. ANALYSIS OF PROBLEM

The classification of the bugs into different buckets can be done using data mining and machine learning concepts. The bugs will be classified into different buckets according to selected features. The approach of bucketing was used by Microsoft for their Windows Error Reporting System [8]. Windows Error Reporting (WER) is a distributed system that automates the processing of error reports coming from an installed base of a billion machines. It collects error data automatically and classifies errors into buckets, which are used to prioritize developer effort and report fixes to users. For Bucketing two types of heuristics were applied: Expanding heuristics increase the number of buckets with the goal that no two bugs are assigned to the same bucket, Condensing heuristics decrease the number of buckets with the goal that no two buckets contain error reports from the same bug. In this we used Naive Bayes Method for Spam Classification explores the use of Bayesian probability approach for text classification. Bayesian is good approach for text classification.

4. PROPOSED WORK

4.1 For Client

4.1.1 Feature Extraction and Selection

A. Overview of the Bug Site

In bug site, bug reports are organized in the form of different attachments and attachments are grouped into General, Commit, Build, Test, Fix Entries category. According to us, attachments of General category are relevantfor classification purpose. General category attachments contain information which is available before the bug is analyzed, tested and fixed by the developer. General category attachments are further divided into Description, Crash info, Decode file, Event log, Email, Static analysis etc attachments. Then bug information is extracted by analyzing the attachments and irrelevant attachments are discarded. For example, Static analysis and Email information etc. are discarded from General category. The information is retrieved from the bug site in html format; html tags are then removed to get individual paragraphs. Information is then statically analyzed to find some pattern for automatic feature extraction.

B. Feature Extraction and Preprocessing

The goal of a bug feature extractor is to automatically extract features from bug information in bug repository after html tag removal. That is, to extract bug information from attachments written in natural language and from the programming language information present in Crash info attachment. Developers and Reporters usually analyze the bugs and points out what causes the bug in natural language in attachments. The Title, Description and Crash file attachments are valuable to us. After analysing bug information, we have find two types of information that can help developers to locate bug. The first type of information is the attributes written in natural language and the other type is the attributes written in programming language. For the attributes belonging to natural language, feature extraction is done using Bag of Word approach. For example, title and description are generally written in natural language, so word frequency information is considered. The words are assigned probability according to their weighting in classification. For example title,” SNMP Query for cempMemBufferMemPoolIndexreturns out of range value” ,” cempMemBufferMemPoolIndex” should be given more weightage than “SNMP Query”. Attributes of programming language include commands, log events and stack trace decode. For programming language type of attributes, first static analysis of the bug information is done to find out the pattern for their retrieval. For example, event log messages will start with % sign and end with colon (:) like %<feature>:. For bug CSCtn56006, the event log contains messages like“arf-server59:2011-03-14T15:45:10:%SCRIPT-6-ETEST:%[pname=TRP-Enhanced_MemPool_MIB]: running script Enhanced_Mempool_Mib_en version 1.7”.

```
*Nov 12 00:30:02.699: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet0/0, changed state to up

*Nov 12 00:30:02.699: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet0/1, changed state to up

*Nov 12 00:30:02.699: %LINEPROTO-5-UPDOWN: Line protocol on Interface VoIP-Null0,
changed state to up

*Nov 12 00:30:03.479: %LINK-3-UPDOWN: Interface Serial0/0/0, changed state to down *Nov
12 00:30:03.479: %LINEPROTO-5-UPDOWN: Line protocol on Interface IPv6-mpls, changed state
to up.
```

C. Feature Selection

We are considering five feature groups: Title, Description, Syslog Event, Commands and Trace Decode. The features contain some noise also. So to reduce the noise the feature selection is performed using Information gain measure. Information gain is a popular score for feature selection in the field of machine learning. In particular it is used in the C4.5 decision tree inductive algorithm, Yang and Pedersen (1997) have compared different feature selection scores on two datasets and have shown that Information Gain is among the two most effective ones[11].

4.1.2 Probabilistic Framwork For Classification

In this paper, we are considering Bernoulli and Multinomial Naïve Bayes Model for bug classification purpose, since Naïve Bayes Model is popular for text classification [12]. A Naive Bayes Classifier can be defined as an independent feature model that deals with a simple probabilistic classifier based on Bayes' theorem with strong independence assumptions [13]. There are several models which assume different fitting for Naïve Bayes. The most common models are: Bernoulli Event Model characterized as Boolean weight which uses binary feature occurrences; another one is the multinomial model which uses feature occurrence frequencies. Consider the bug classification into n different classes C = {C1,C2…….,Cn}.The unseen bug(Bi) will be classified using to class with higher posterior probability. Instead of taking word information as input we are using feature information for bug specific features and for features of natural language type we are considering word information. Words are unigram features but extracted features from bug information may be Bi-gram, Trigram or Multigram. Bug specific features may be a combination of number of words as in Trace Decode and Commands.

A. Bernoulli Event Model

In the Bernoulli event model, a bug is represented as a binary vector over the space of features. BVi is the feature vector for the ith bug Bi. We have a vocabulary V containing a set of |V| features each dimension t of the space, where t Ɛ {1,2……|V|}. Dimension t of a bug vector corresponds to feature Ft in the vocabulary. BVit, is either 0 or 1 representing the absence or presence of feature Ft in the ith bug. With such a bug representation, we make the naive Bayes assumption: that the probability of each feature occurring in a bug report is independent of the occurrence of other features in a bug.

B. Multinomial Event Model

In contrast to the Bernoulli event model, the multinomial model captures feature frequency information in bugs. Consider, for example, Mi is the multinomial model feature vector for the ith bug data Bi. Mit, is the number of times feature Ft occurs in bug data Bi; ni= ∑t Mit the total number of features in Bi. In the multinomial model, a bug is an ordered sequence of feature events, drawn from the same vocabulary V. We assume that the lengths of bugs are independent of class. We again make a similar naive Bayes assumption: that the probability of each feature event in a bug is independent of the feature's context and position in the document. P(Ft|Ck) is estimated using word frequency information from the multinomial model feature vectors. Generation of bugs is modeled by repeatedly drawing features from a multinomial distribution.

4.1.3 Dataset Information

The data from six different categories os, bgp, ip, ipv6, aaa, snmp are collected from the site of one of the networking based organization. The training data contains 1000-1500 bugs from each category. Five different types of features are extracted by static analysis and pattern matching. The Syslog Event contain around 30,000 Syslog messages, for Commands The vocabulary size is around 600 commands, for Title and Description word frequency data is taken and their respective vocabulary sizes are around 9000 and 30000. For Trace decode, there are 400 chunks available for the classification purpose.

4.1.4 Performance Measure

In classification system, the terms true positives, true negatives, false positives, and false negatives are used to compare the results of the classifier under test with known external output. The terms positive and negative refer to the classifier's prediction and the terms true and false refer to whether that prediction corresponds to the external output. To evaluate the performance of bug classification system, we are using four standard measures Precision, Recall, Accuracy and F-Measure. The Precision and Recall will be calculated for all the categories (class) and Accuracy is taken to assign the Precision and Recall values for the classification.

4.2 For Admin Tester

4.2.1 Admin Module

The administrator too has the authority to update the master details of severity level , status level, etc, modules of the project. The administrator adds the users and assign them responsibility of completing the project. Finally on analyzing the project assigned to the particular user, the administrator can track the bugs, and it is automatically added to thetables containing the bugs , by order of severity and status.

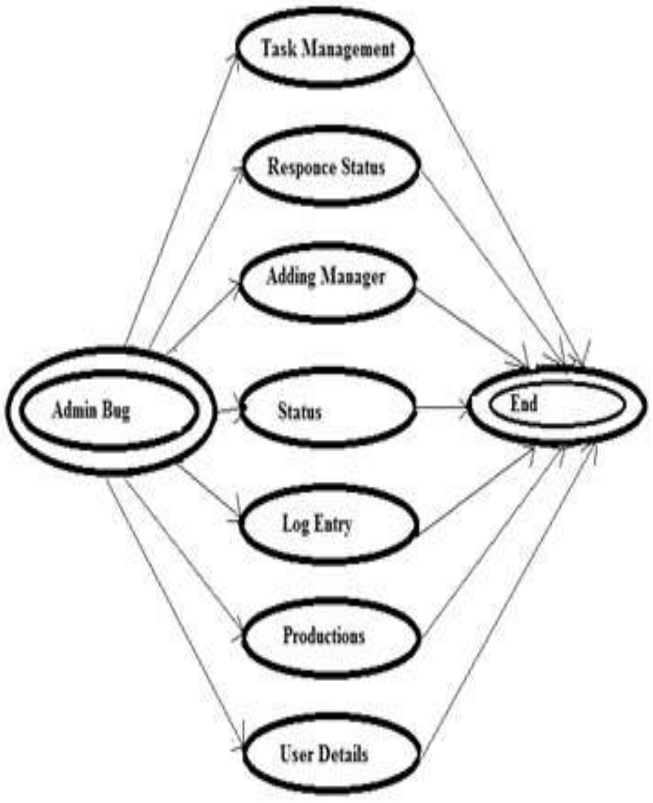


Fig-1: Admin module in Bug Tracking System

4.2.2 Manager Module

The administrator can know the information in tact the various projects assigned to various users, their bug tracking status, their description etc in the form of reports from time to time. The project 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.

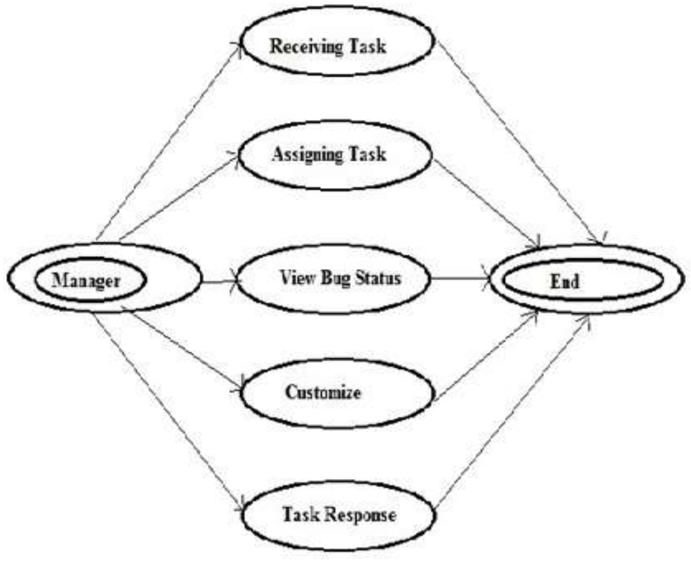


Fig-2: Manager module in Bug Tracking System

4.2.3 Developer Module

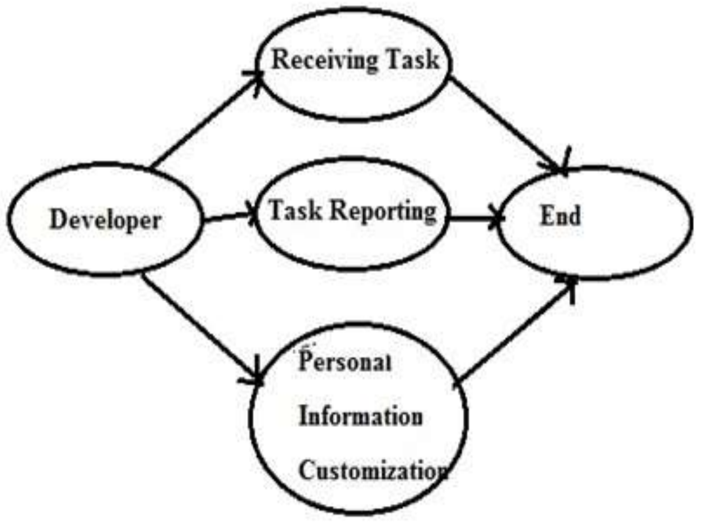


Fig-3: Developer Module in Bug Tracking System

4.1.4 Structure Flow Diagram

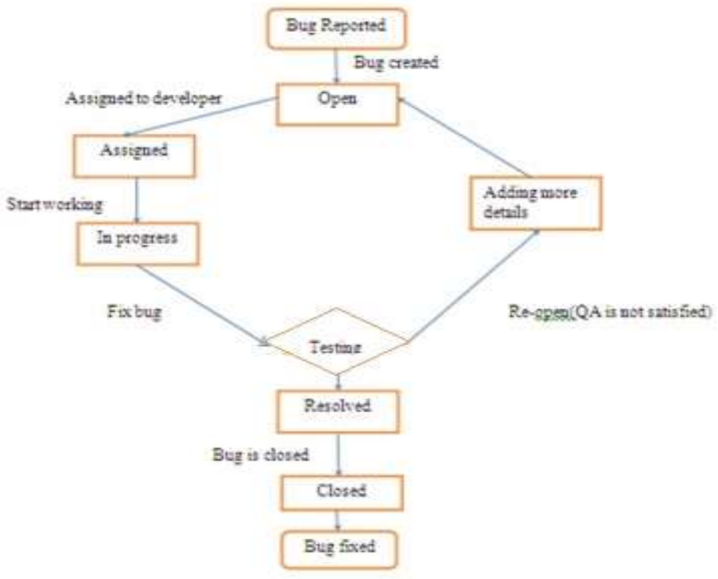


Fig-4: Admin Side structure flow

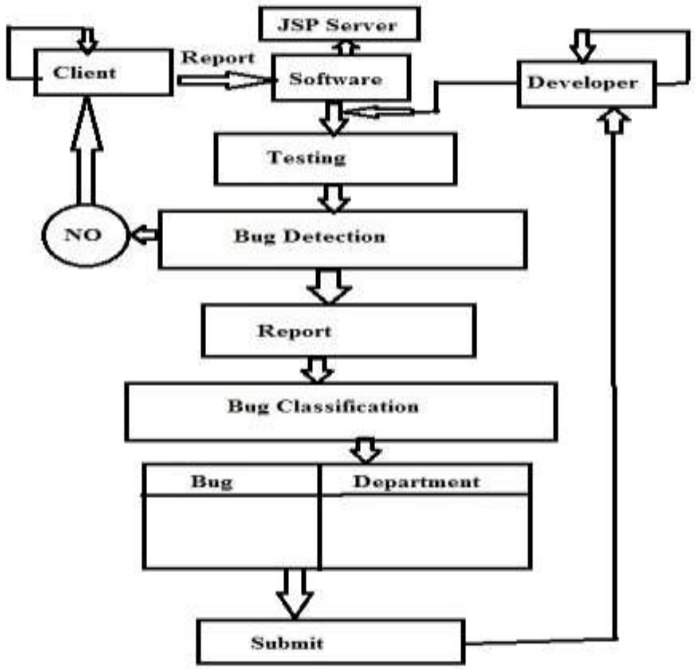
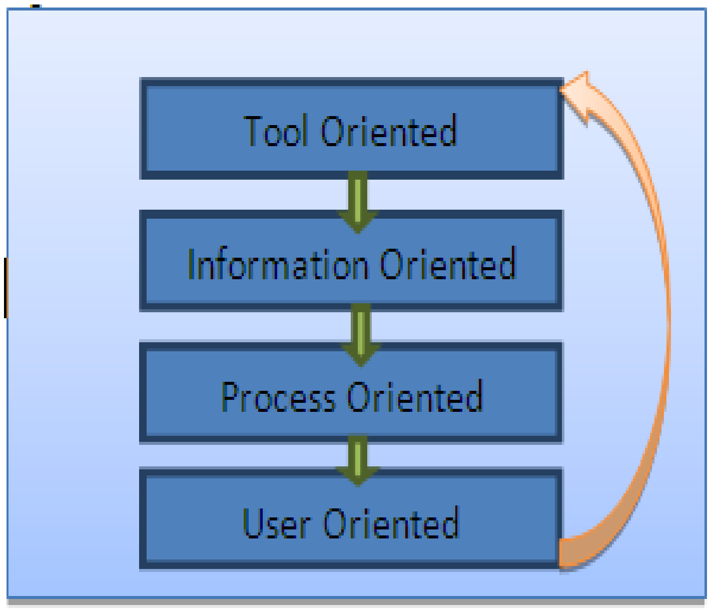


Fig-3: Structure Flow of Both Client And Admin Side

4.2.5 Effective Bug Tracking Systems

Software engineers often involve in fixing bugs in the system under development. The time they spend on that can be reduced and the quality of software can be increased by using an effective bug tracking system. The initial information on a bug can help the engineers to resolve bugs faster thus saving development time and cost. Over a period of time development team can reduce lot of wastage of time with regard to tracking and fixing bugs with a sophisticated bug tracking system in hand. This fact motivates us to build a conceptual framework that provides required directions that can improve bug tracking systems. Out proposed framework is as given in fig. As can be seen in fig., the directions proposed are tool oriented, information oriented, process oriented and user oriented in nature. They are iterative directions that can be cycled back after each iteration.



Fig(f). Proposed Directions to Enhance Bug Tracking Systems

Tool oriented

We recommend a bug tracking system to be tool oriented. Tool oriented does mean that the bug tracking systems can be configured to collect stack trace implicitly and add it to the report that contains bug details. It can improve the information collection capabilities by doing so. It can also use steps to make use of macro recorders, capture/replay tools that can be observed by software engineers later. From the observations, it is possible to capture test cases and fix the bugs easily. Tool oriented feature can enhance the capability of bug tracking system in terms of information collection that is quite relevant and can be readily used to fix bugs in the system. This will lead the effective tracking and fixing of bugs those results in quality of the software besides being productive. This direction can helps in transition into information oriented

Information oriented

This is another direction from us that helps software developers to have improved focus on the collection of information that has to be kept in bug reports. Towards this end some sort of tools like Cuezilla can be embedded into bug tracking systems. Such tools verify the information provided in bug reports and provide real world feedback that helps improve the quality of information. This will help software engineers to get motivated and have more focused work in solving or fixing bugs. Being information oriented with tool support can enhance the possibilities of checking whether the stack traces reported are complete and consistent in the given scenario. This direction when followed by real world bug tracking systems can lead to make it more process oriented.

Process oriented

This direction is meant for process oriented improvements. The process being followed in bug tracking systems can be improved further using this direction. Is does mean that all administrative activities pertaining to bug tracking and fixing come under process oriented feature. The process oriented bug tracking systems can also focus on the developer who is made responsible to fix bugs besides having a comprehensive bug report. Other advantages of process oriented feature are that developers can have better awareness on bug reports and thus they are aware of possible actions to be taken; it also helps developers to estimate time to be spent on specific bugs and schedule their time accordingly. Process oriented feature can influence user-oriented feature.

User oriented

Users do mean the developers and also bug reporters. This direction focuses on educating the reporters so as to enable them to collect correct information and how to collect it as well. This training helps both developers and reporters. The expected information in the bug report makes the developers to grasp it faster and act quickly to fix bugs in real time applications. This direction can influence the adaption of new tools in the process thus making it more robust and productive in nature. To support this we have built a prototype application that makes use of all directions and enhance the capabilities of bug tracking systems in the real world.

5. APPLICATION

Bug classification is entirely different from text classification. The classes which are considered in bug classification (all bugs of network type) are on the basis of network protocols like IPV4 (referred as IP in literature), IPv6, SNMP, BGP and OS. Since these classes have many common characteristic and do not have fixed boundaries like what we have for text classification. The feature selection has been done on the basis of Information Gain Measure. Experimental results for two class classification using word information taken from the bug data gives an accuracy of around 60% and 78% for Bernoulli and Multinomial respectively. But as we move to Multiclass classification, it gives an accuracy of less than 15%. According to us, using bug specific features like Title, Description, Syslogs, Commands and Trace Decode accuracy of classification can be increased. The experimental results show the effect of applying Bernoulli and Multinomial Model to the bug data using bug specific features. We can observe that Bernoulli Model is giving good results when compared to the Multinomial model. And for Trace Decode the results are being similar. After assignment of priorities to the feature groups, the feature groups are arranged. The new unseen bug will go through the feature group checking in the order of the priorities assigned to them. At the time of classification considering Description and Syslogs feature groups, Multinomial model is applied. For rest of the feature groups Bernoulli model is referred. The overall accuracy of the classification is found to be 55% after applying the mentioned sequence.

6. CONCLUSION

The experimental results show that applying Bayesian model for classification of bugs using word information as feature is reliable for two classes. It can be concluded from the results that we need to go into semantics of bug information. We had successfully extracted some of the bug specific features. According to the results, Bernoulli and Multinomial Models using bug specific features give better accuracy compared to word information

7. REFERENCES

1. E. H. Miller, “A note on reflector arrays (Periodical style—Accepted for publication),” IEEE Trans. Antennas Propagat., to be published.

2. J. Wang, “Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication),” IEEE J. Quantum Electron., submitted for publication.
3. C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.
4. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interfaces(Translation Journals style),” IEEE Transl. J. Magn.Jpn., vol. 2, Aug. 1987, pp. 740–741 [Dig. 9th Annu. Conf. Magnetics Japan, 1982, p. 301].
5. M. Young, The Techincal Writers Handbook. Mill Valley, CA: University Science, 1989. [10] J. U. Duncombe, “Infrared navigation—Part I: An assessment of feasibility (Periodical style),” IEEE Trans. Electron Devices, vol. ED- 11, pp. 34–39, Jan. 1959.
6. S. Chen, B. Mulgrew, and P. M. Grant, “A clustering technique for digital communications channel equalization using radial basis fuction networks,” IEEE Trans. Neural Networks, vol. 4, pp. 570–578, July 1993.
7. G. R. Faulhaber, “Design of service systems with priority reservation,”in *Conf. Rec. 1995 IEEE Int. Conf. Communications,* pp. 3–8.