International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 9 • Number 46 • 2016

Mining Algorithms Precedence List for Software Bug Classification

Gitika Sharma[a] and Sumit Sharma[b]

[a,b]*Department of Computer Science and Engineering, Chandigarh University, Ghruan, Punjab, India. Email: [a] gitikasharma41@gmail.com*

Abstract: Software development methodologies have changed from static Waterfall Models to highly dynamic Rapid Application Development model; but the need to resolve bugs in the most efficient manner has remained constant. This lead towards the need of systems that can help to effectively record the changes in the states of the open bug as well as effectively aligning them to the most suitable developer, commonly known as Bug Triaging Systems (BTS). This paper aims at providing support in the development of Ideal BTS by creating a precedence list of various mining algorithms which are used in Software Bug classification. Hence, using the precedence list the BTS creators can enhance the capabilities.

Keywords: Bug, Bug reports, Bug Tracking System, Bug Life Cycle, Bug Triaging, Feature Selection, Machine Learning Algorithm.

1. INTRODUCTION

Software repositories contain important information about software projects. It is a vital component in modern software development. Many software projects create and maintain bug repositories for effective and efficient bug tracking and controlling [1].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. Development of bug tracking system has targeted one of the most important repositories that are bug repository.

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[2]. Example of such bug repository is Eclipse bug repository (https://bugzilla.Eclipse.org).

Now a days many open source software provide Bug Tracking System along with it that keeps track of the reported bug. “Bugzilla”, “jira”, “FogBugz”, “Fossile” and “ikiwiki” are some example of bug tracking systems[3].

Users can file bug in this tracking system and this bug reports are sent to developers for fixation. To report a bug in bug tracking system, a user is asked to fill a form that provides necessary details about the bug. Fundamental testing principles of bug reporting systems are:

1. Report the bug as soon as it is found.
2. Describe the bug in effective manner- Bug report should be precise and concise but clearly state about the problem encountered. There should be good attachments like snapshots or narrate a story that tells how bug was produced so that one should be able to reproduce the bug.
3. Do not be judgmental while reporting the bug- Bug report should target on the problem encountered but not on the developer.
4. Follow up your bug report- Bug report should be followed up and also give necessary comments when it is required.

This form has one-line summary of the failures which are observed and particular component in which that bug occurs. Important elements of bug reports are:

i. Reporter's Name or ID-One who encounter bug and file report to fix it.

ii. Type of change request - It states the type of bug report i.e. if the reported bug is new enhancement, bug or new requirement. User fills this element when he/she report the bug.

iii. Bug Id- It is unique id of bug that distinguishes it from other bug. It is automatically generated when bug is filed.

iv. Description- A brief description of the problem is stated in this column. User is asked to fill this attribute at the time of reporting bug. This description must be clear to understand e.g. "Switching to "Design" mode in SWT closes Eclipse." is brief and good example of description of bug.

v. Component- Component where bug is found, should be specified. It helps in assigning the bug to a developer who is assigned to that component of the product.

vi. Resolution- This field describe that whether bug is fixed or not. This element is filled and updated by developer when bug is assigned to him.

vii. Status- This field specifies the present status of the bug e.g. NEW, RESOLVED. This element is set by project manager when a bug is reported or when bug all necessary changes are made to resolve the bug.

viii. Create Date- This field is filled when bug is reported. It tells the date when the bug was reported.

ix. Date of Close- This field is set by project manager, it tells the date when the bug was closed.

x. Version number- This field indicates the version number of software in which bug was found. This is important part of information that tells that if the bug was found in shipping version or in build under testing and development. It is filled at the time of reporting bug by reporter of the bug.

xi. Operating system and Platform- This field is filled by bug reporter, it depicts the environment on which the bug was occurred.

xii. Attachment - This field tells the number of attachment that was uploaded by reporter of bug that can help developer to understand the bug and reproduce it.

xiii. CC List - A user can add himself in cc list if he finds the bug encountered by him is already reported. Users in cc list also get mail when any progress is made related to that bug.

xiv. Bug Fixed Time - It is time taken to fix the bug i.e. difference between the time when the bug was opened first time to the time when the bug was closed. This field is filled by project manager.

xv. Priority- This field describes the importance of bug from other bug and tells the order in which bugs should be fixed. Users fill this element when he files the bug report. There are five levels of priority from P1 to P5. P1 is having highest priority and P5 is having least priority.

xvi. Severity- Severity of bug depicts the impact, it is filled by user when he reports the bug. This field is used by developer for classifying bug. In Table 1, description of severity of bugs that is used in Bugzilla (BTS) is described:

Table 1
Description of Severity

Severity	*Description*
Blocker	Blocks development and/or testing work, production could not run
Critical	When there is memory leak, crash or data loss
Major	When there is major loss of function
Minor	When there is minor loss of function
Normal	It is default option in bug tracking system for bug severity
Trivial	These are cosmetic problem e.g. Wrong alignment of text or wrong spelling of words.
Enhancement	It is request for new enhancement

These elements of bug reports are not only filled by users but project manager or developer also fills some of the element of bug report at different time and phases of bug life cycle. User guidelines are provided for the information about how to fill each field of bug report so that bug reports produced are consistent and of good quality. Bug goes through different phases during its life cycle. The cycle starts when bug is reported and ends when that bug is closed and never reproduced.

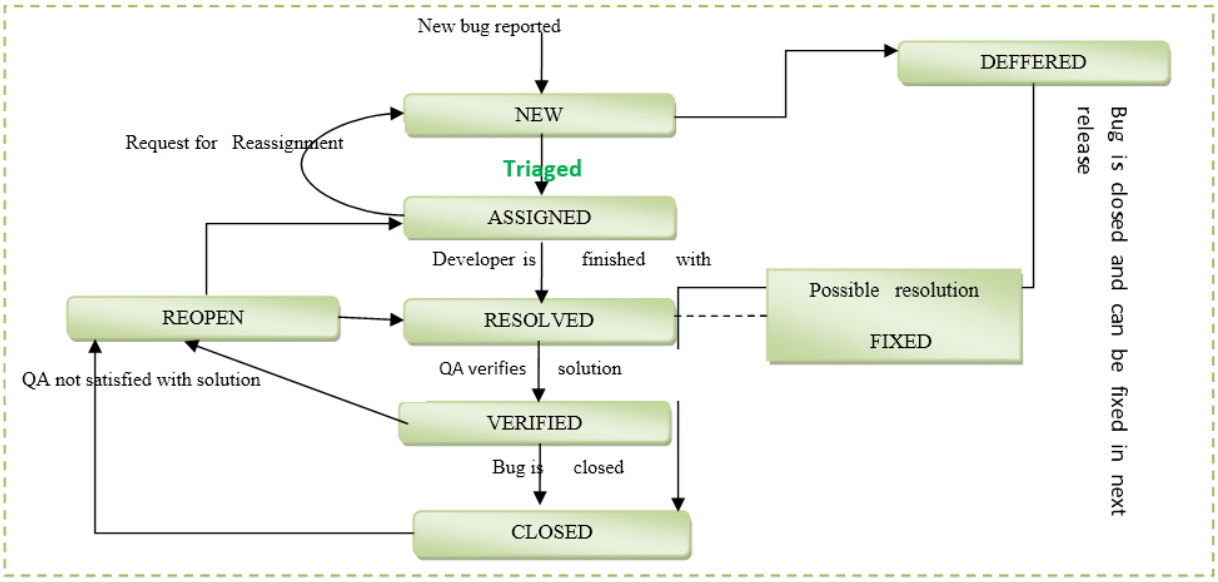

Figure 1: Generic Bug Life Cycle

Various BTS have chosen their own terminologies for the different phases of the bug life cycle. The Figure1 shows the generic bug life cycle. When a new bug is reported, the bug is analyzed to check its validity

and importance. If the reported bug is valid then only bug triaging task is performed otherwise it is assigned deferred status

Bug Triaging is the task that is performed to evaluate the bug and to decide when the bug is considered as resolved and to whom this bug should be assigned. Bug is assessed based on its severity and priority, the complete bug triaging task is shown in Figure 2.

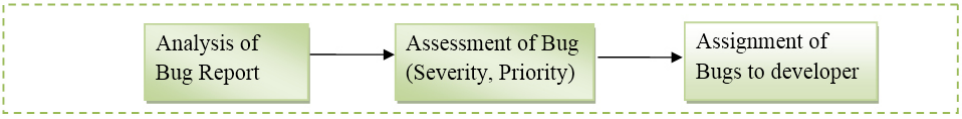

Figure 2: Bug Triage Process

Bug severity indicates the impact of the reported bugs i.e. which bug may act as a showstopper and which bug may simply be put in the deferred state It helps Traiger to triage the reported bugs, Triager is person who triage the bugs manually based on his past experience and knowledge. But, it is observed that there are large numbers of bugs reported on the daily basis [4]. So, the task of triaging the bugs has become tedious and cumbersome. There is need to put a lot of effort to manually assign the severity to bugs and then assign the bug reports to developer. The solution of this problem could be to automate the triaging process that can save a lot of effort and time. Additionally, the accuracy of triaging process depends on triager's knowledge and past experience that vary from person to person, but automation of this task could improve the accuracy level of this task as well.

In past, many researchers have made an attempt to automate the triaging task, and still there is scope of improvement in this domain. This paper focuses on automation of the severity classification of the bug report which further aids to automate the triaging task. It is based on the hypothesis that there are terms used in summary of bug report that are called severity terms which could help in identifying the severity level of the bug reports. Dictionary of these terms are created with help of feature selection algorithms and then supervised machine learning approach is used for predicting the severity. The number of terms that are required to train classifier are analyzed and selection of these terms are done by different feature selection algorithms and then precedence list of mining algorithms are determined to find out which machine learning algorithm gives higher performance than other.

The rest of the paper is organized as follows: Section II discusses related research work. The proposed algorithm of severity classification is provided in Section III. Section IV presents experimental set up and results of the proposed solution is discussed in Section V. Section VI discusses the conclusion of the followed approach and the future scope.

2. LITERATURE SURVEY

Bug triaging is task that is performed when a new bug is reported. But as past studies shows that large number of bugs are reported, so it becomes time consuming.

In 2004, Davor Cubanic, Gail C. Murphy[5] made their first attempt to use ML algorithms to help triaging task. Their work was extended by John Anvik et. al.[6], they used some different supervised ML algorithms such as C 4.5 Support Vector Machine (SVM), Naïve Bayes(NB) and achieve precision level around 60%. John Anvik[7] further extended his previous work and creates recommender for recommending developers to whom bug reports should be assigned.

Tim Menzies et. al.[8] proposed a method called SEVERity ISsue assessment (SEVERIS) to automatically assign severity level to bug reports using text mining and machine learning techniques. An automatic routing

method was proposed by Giuliano Antoniol et. al.[9] that was used to route the real bugs for maintenance and enhancement requests to project leader automatically. Ahmed Lamkanfi and co authors [10-11] proposed a new approach for classification of bug reports of open source software on severity basis. Yuan Tian et. al. [12] proposed to use BM25 similarity function to find severity level of bug reports. Akinori Ihara et. al.[13] proposed a model called bug fix time prediction model. This model predicts the bug fix time i.e. time by which bug will be fixed by developer after it has been reported.

Jin-woo Park et. al. [14] discussed the problem of sparseness associated with recommendation systems proposed in literature for bug triaging system. Further authors proposed an algorithm called COSTRIAGE. This algorithm can help to overcome the problem of sparseness in recommender system. Meera Sharma et. al. [15] proposed a prediction model to predict cc list. This list contains the man power participated in fixing bugs. Marcelo Serrano Zanetti et. al. [16] proposed a method that identifies which bug reports are valid. Pamela Bhattacharya et. al.[17] presented an empirical study to analyze the bug fix process in android application. Mamdouh Alenezi et. al. [18] presented an approach to automate the process of assigning bug reports to appropriate developer. Term selection method was used to choose discriminating terms to classify bugs. Naïve bayes classifier was used to create predictive model.

Cheng-Zen Yang et. al. [19] made an investigation to find the influence of four quality indicators of reported bug in predicting severity of bug rather than using textual description alone.Kanwal et. al. [20-21] proposed an approach to prioritize the bug reports. Further performance comparison of Naïve bayes (NB) and Support vector machine (SVM) was done. Meera Sharma et. al. [22] proposed an approach to predict priority of bug reports. The approach was applied on intra and cross projects. The experiment was conducted over Eclipse project and OpenOffice datasets. Different machine learning algorithm was applied such as Naïve bayes, KNN, SVM, Neural network to predict priority. Shruti Gujral et. al.[23] proposed a solution of severity classification by creating dictionary of terms. The same work was extended by Gitika Sharma et. al. [24] to classify bug reports based on severity using feature selection method and supervised approach.

The proposed algorithm to solution of the automation of severity classification task is provided in next section. Different feature selection approach and ML algorithms are applied and comparison of these algorithms is performed.

3. PROPOSED ALGORITHM TO SEVERITY CLASSIFICATION

The automated system developed for severity classification has mainly five steps. These steps of algorithm are explained as follows:

Algorithm for Automated Severity Classification of Reported Bugs

Argument list: Bug report instances, no of cross fold validation

Returns: Prediction of severity of bug report instances

1. **Data Acquisition:**

   (a) The required fields of bug reports i.e. bug id, summary, severity field is extracted from bug database and other bug fields are ignored.

2. **Pre-processing of bug reports:**

   (a) Tokenization- The summary of bug reports are tokenized by removing punctuation marks, symbols, brackets and hyphens etc.

(b) Stop word algorithm- In this part all the unnecessary words like articles, prepositions, conjunctions and adjectives are removed.

(c) Stemming - Stemming is to reduce the word to its root.

(d) Feature vector model- The word that is obtained in previous step is used as feature with its associated weight. The weight of these features is calculated using Term frequency and Inverse document frequency (TF-IDF).

3. **Feature Selection:**

(a) In this work different feature selection methods are used to select the feature that could serve as best indicator of bug severity.

4. **Dictionary of Terms:**

(a) Top *k* terms after apply feature selection are used to create dictionary of severity indicator terms

5. **Classification:**

(a) Classification of bug reports are performed using supervised machine learning algorithms such Naive bayes, naive bayes multinomial, *k*-nearest neighbo (K-NN) and Support vector machine (SVM).

6. **Performance Evaluation:**

(a) K-fold cross validation is used to validate the process. Performance parameter such as accuracy and precision is computed to compare performance.

4. EXPERIMENTAL SETUP

In this section, experimental setup is provided in detail. First dataset used for the experiment is explained and then all the pre-processing steps, feature selection algorithms and classification algorithms are explained. Afterwards, performance measures used to evaluate the experiment is explained.

4.1. Dataset Acquisition

This experiment considers instances of bug reports of four components of Eclipse. These components are core, UI, debug and SWT. Bug report instances of Eclipse are downloaded from bug repository. These instances were reported in Bugzilla (Bug Tracking System)[25]. The instances have severity of type blocker, critical, Enhancement, major, minor, normal, trivial. The report instances that have normal and enhancement severity type are not considered in experiment. The reason for the exclusion of normal severity is because it require manual inspection to assess the severity level of these bug report as it represents grey zone[26]. Enhancement is request to team lead to add new feature and do not represent a real bug. Severity level is classified into binary levels. Bug reports of critical, blocker and major severity levels are taken as severe bug report while minor and trivial are taken as non severe bug reports. The dataset used in the experiment is given in Table 2.

Eclipse is an integrated development environment (IDE) that is used worldwide for the development of software. Thus, the users of Eclipse are developer themselves and bug reports should be of high quality [27]. It could help in our approach to create dictionary of severity indicator terms.

Table 2
Bug report instances of Eclipse

Component of Eclipse	*Severe Bug report*	*Non Severe Bug report*
Core	1514	1487
Debug	1514	1400
SWT	1618	1393
UI	1764	1432

4.2. Preprocessing Steps

It is used to distill unstructured data to structured format. There are different preprocessing steps performed in Text mining such as tokenization, stop word removal and stemming. These algorithms are discussed below.

4.2.1. Tokenization

The purpose of tokenization is to remove all the punctuation marks like commas, full stop, hyphen and brackets. It divides the whole text into separate tokens to explore the words in document.

4.2.2. Stop Word Removal

The purpose of this process is used to eliminate conjunction, prepositions, articles and other frequent words such as adverbs, verbs and adjectives from textual data. Thus it reduces textual data and system performance is improved.

4.2.3. Stemming

Stemming is used to reduce the words to their root words e.g. words like ”computing”, ”computed” and “computerize” has it root word “compute”. The purpose of stemming is to represent the words to only terms in their document. There are different algorithms to perform stemming such as Lovins Stemmer [28], Porters Stemmer [29], Paice/Husk Stemmer [30], Dawson Stemmer[31], N-Gram Stemmer[32], YASS Stemmer[33] and HMM Stemmer[34].

4.2.4. Weighting Factor

Features are extracted from overloaded large datasets.TF-IDF (Term frequency- Inverse document frequency)[35] score is generally is used to give weight to each term. TF-IDF is multiplication of term frequency and inverse document frequency.

4.2.5. Term-Document Matrix

After initial steps of preprocessing text in documents is used to formulate term- document matrix. Rows in matrix represents document in which word appears and columns represent the words that are extracted from documents.

The above mentioned preprocessing steps are performed on bug report dataset and Term document matrix is obtained where rows represent bug id and columns represent term. The cell of matrix is filled with TF-IDF score.

4.3. Feature Selection

It is the process in which subsets of terms are selected from the training datasets. The selected subset of feature set is only used for classification task. The first goal of feature set is to increase efficiency of classifier by decreasing

the size of training datasets as it could be more expensive in terms of time required to train classifier on large datasets. The second goal of feature selection is to eliminate noise feature. A noise feature in training datasets is the one that increases the classification error. Thus, feature selection can be viewed as a method that replaces complex classifier (classifier that uses all features) with a simpler classifier (reduced feature set). There are different feature selection methods used in this approach such as info-gain, Chi square, Gini Index, correlation and principle component Analysis (PCA).

4.4. Dictionary of Terms

The terms are sorted in descending order according to weight assigned to them by feature selection method. Then top *m*-terms are used for creating dictionary of terms that could serve as severity indicator of bug reports.

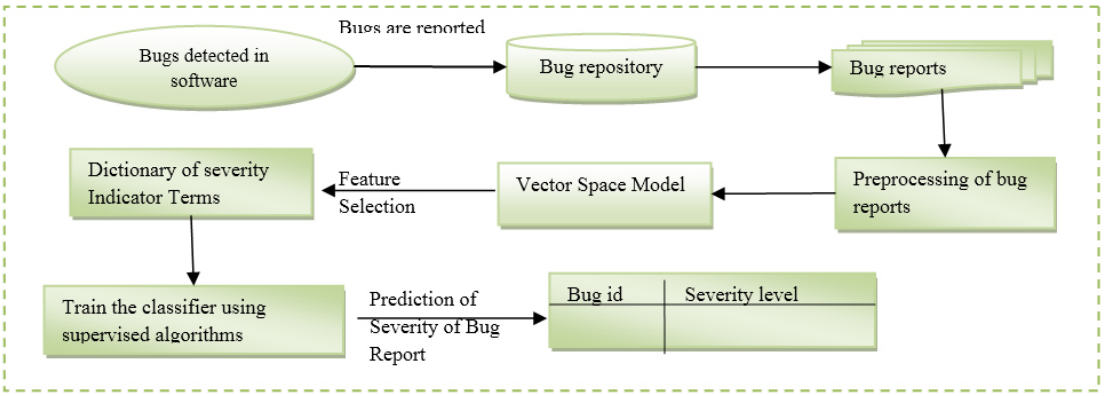

Figure 3: Proposed Architecture of Severity classification of bug reports

4.5. Supervised Machine Learning Algorithm

In supervised learning, the instances are labelled with known classes labels. In this scenario, the dataset knows the target class before classification. Thus, it is very helpful for the problems which have known inputs. Different supervised machine learning algorithms such as Naive Bayes, Naive bayes multinomial, Support vector Machine and K-nearest neighbors are used in the experiment. Rapid miner [36] is used to train the classifier in the experiment. 5- Fold cross validation approach is used to validate the results.

4.6. Performance Measurement

The performance of experiment is evaluated using performance measure such as Accuracy.

5. RESULTS

The results of the proposed approach are presented in this section. The numbers of terms are varied from 25 to 200 to know the optimal number of terms that are required to feed the classifier for better results. The selection of these terms are done by using different feature selection algorithms such as Chi square, info gain, PCA, Gini Index and correlation. Different machine learning algorithms such as Naïve bayes, Naïve bayes multinomial, SVM and K-NN are evaluated on the basis of accuracy to find their precedence list for the bug report classification. The results obtained with different algorithms are explained below.

5.1. Naïve Bayes Algorithm

The performance of naïve bayes algorithm for different component of Eclipse is shown in Table 3:

Table 3
Accuracy of Naïve Bayes algorithm

Component	*Feature Selection Algorithm*				
Core	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	62.23	64.5	59.93	64.33%	65.23%
50	65.17	67.47	62.77	67.23%	67.17%
75	66.9	68.03	63.7	68.30%	67.43%
100	68.77	68.9	67.14	68.90%	66.87%
125	68.37	67.33	67.67	67.90%	66.30%
150	68.27	66.43	69.33	66.83%	66.60%
175	66.2	66.13	66.73	66.17%	66.43%
200	66.27	66.27	66.27	66.27%	66.27%
SWT	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	64.82%	63.22%	61.78%	67.64%	68.04%
50	68.45%	65.38%	65.51%	66.26%	67.20%
75	70.52%	65.13%	66.32%	69.86%	67.89%
100	71.08%	65.98%	68.08%	68.54%	64.91%
125	71.83%	65.26%	69.61%	66.98%	65.01%
150	70.39%	64.32%	68.70%	66.13%	65.85%
175	69.83%	64.48%	69.14%	66.13%	64.41%
200	69.55%	64.63%	69.11%	64.88%	61.10%
UI	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Corelation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	60.00%	64.58%	50.59%	63.81%	69.75%
50	69.41%	63.22%	55.34%	71.02%	65.68%
75	68.14%	66.86%	58.81%	70.93%	67.29%
100	67.71%	65.51%	59.15%	69.92%	67.29%
125	68.31%	66.61%	62.63%	68.05%	67.03%
150	66.44%	64.32%	60.25%	66.02%	67.29%
175	65.00%	63.90%	59.92%	65.08%	64.75%
200	64.24%	63.31%	57.54%	62.88%	63.22%
Debug	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	65.78%	67.21%	56.69%	67.77%	67.41%
50	67.87%	70.73%	64.29%	69.10%	72.02%
75	69.76%	71.69%	67.24%	73.18%	71.06%
100	70.83%	69.56%	66.05%	70.46%	69.40%
125	69.36%	68.63%	66.68%	68.63%	67.21%
150	70.13%	68.63%	64.98%	68.63%	63.86%
175	69.50%	68.60%	65.12%	68.47%	63.86%
200	63.82%	63.82%	63.82%	63.82%	63.82%

The best accuracy is achieved using Chi square feature selection method and it is observed that approximately 100 terms give maximum results in terms of accuracy.

Naïve bayes multinomial: The performance of naïve bayes multinomial algorithm for different component of Eclipse is given in Table 4:

Table 4
Accuracy of Naïve Bayes multinomial algorithm

Component	*Feature Selection Algorithm*				
Core	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	65.17%	65.17%	62.03%	65.43%	66.00%
50	68.37%	69.47%	63.77%	69.23%	69.13%
75	69.33%	71.17%	66.63%	70.77%	71.83%
100	70.57%	72.63%	67.63%	72.57%	71.97%
125	70.03%	72.33%	69.33%	72.40%	73.37%
150	71.60%	71.57%	70.73%	72.10%	72.40%
175	71.73%	71.87%	71.70%	71.87%	72.03%
200	71.77%	71.77%	71.77%	71.87%	71.87%
SWT	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	64.44%	66.17%	59.87%	65.88%	65.48%
50	65.54%	67.29%	63.85%	67.29%	68.33%
75	66.38%	68.86%	64.82%	68.33%	69.11%
100	67.07%	69.67%	65.82%	69.30%	69.67%
125	68.04%	69.67%	67.57%	69.51%	69.77%
150	68.58%	70.05%	67.01%	69.92%	70.83%
175	68.45%	69.98%	67.07%	70.30%	70.74%
UI	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Corelation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	61.44%	62.12%	60.51%	62.20%	62.37%
50	62.71%	63.56%	60.59%	63.22%	63.98%
75	62.97%	64.32%	61.19%	64.07%	64.15%
100	63.39%	64.07%	61.78%	63.73%	63.98%
125	63.14%	63.39%	62.03%	63.39%	64.41%
150	63.31%	63.64%	62.46%	63.64%	63.90%
175	62.88%	63.90%	62.63%	64.24%	64.07%
200	63.64%	63.90%	62.71%	64.07%	64.07%
Debug	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	64.16%	66.51%	60.44%	66.05%	67.14%
50	67.91%	69.50%	62.66%	70.53%	70.99%
75	69.60%	71.79%	66.31%	72.09%	71.22%
100	70.33%	71.13%	68.37%	71.06%	72.19%

Component	*Feature Selection Algorithm*				
125	70.59%	71.66%	70.06%	71.72%	72.05%
150	70.26%	71.59%	70.89%	71.22%	71.89%
175	71.26%	71.29%	71.09%	71.19%	71.29%
200	71.29%	71.29%	71.29%	71.29%	71.29%

In NBM classifier the best accuracy in all components is achieved using approximately 200 terms with all the feature selection methods.

Support vector Machine: The performance of support vector machine algorithm for different component of Eclipse is provided in Table 5:

Table 5
Accuracy of Support Vector Machine

Component	*Feature Selection Algorithm*				
Core	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	62.97%	64.10%	64.03%	64.03%	64.47%
50	64.67%	65.90%	66.43%	66.43%	66.27%
75	67.93%	67.63%	67.90%	67.90%	67.47%
100	69.40%	70.00%	70.13%	70.13%	67.57%
125	70.33%	70.73%	70.80%	70.80%	69.03%
150	71.73%	70.17%	70.73%	70.73%	71.13%
175	71.77%	71.80%	71.80%	71.80%	71.07%
200	71.67%	71.67%	71.67%	71.67%	71.67%
SWT	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	64.57%	65.95%	62.88%	63.44%	66.67%
50	67.82%	68.04%	65.01%	67.17%	67.92%
75	68.08%	68.79%	66.67%	69.11%	68.92%
100	69.64%	70.89%	67.76%	70.20%	69.64%
125	70.14%	71.42%	69.61%	70.67%	70.39%
150	70.36%	71.61%	69.77%	71.36%	71.46%
175	70.64%	71.39%	70.30%	71.71%	71.17%
200	71.42%	71.64%	70.30%	71.52%	71.46%
UI	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	66.27%	66.69%	63.14%	66.69%	64.24%
50	65.59%	66.69%	65.00%	66.53%	67.03%
75	65.76%	69.15%	65.85%	67.71%	68.98%
100	65.76%	70.93%	65.25%	71.10%	70.76%
125	67.80%	69.92%	65.42%	69.41%	70.76%
150	68.22%	70.34%	66.02%	70.42%	70.68%
175	68.22%	70.93%	65.93%	70.25%	70.17%
200	68.90%	70.08%	66.53%	69.83%	69.41%

Component	*Feature Selection Algorithm*				
Debug	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	64.55%	64.39%	60.67%	64.92%	65.08%
50	64.92%	66.31%	62.33%	67.01%	67.34%
75	66.61%	69.07%	65.25%	67.74%	67.31%
100	67.71%	69.80%	68.60%	69.63%	69.13%
125	69.30%	69.40%	69.47%	70.16%	70.86%
150	69.86%	70.56%	70.20%	71.32%	72.19%
175	71.66%	71.82%	72.65%	71.72%	71.49%
200	72.95%	72.95%	72.95%	72.95%	72.95%

Here, the best results are obtained using approximately 175 terms with all the feature selection methods.

K-nearest neighbor: The accuracy obtained using 5-NN classifier with different feature selection method is shown below:

Table 6
Accuracy of KNN algorithm

Component	*Feature Selection Algorithm*				
Core	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	67.37%	63.97%	64.93%	64.00%	64.00%
50	71.63%	72.00%	69.13%	72.13%	71.27%
75	73.17%	73.83%	72.07%	73.60%	73.90%
100	73.10%	74.83%	71.33%	74.60%	75.07%
125	72.40%	73.37%	72.63%	73.83%	73.97%
150	74.47%	73.07%	73.67%	73.50%	74.73%
175	74.27%	74.53%	74.33%	74.53%	74.40%
200	74.37%	74.37%	74.37%	74.37%	74.37%
SWT	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	63.41%	65.88%	63.26%	63.63%	63.82%
50	70.36%	69.86%	66.73%	69.20%	69.77%
75	72.49%	69.08%	69.70%	70.89%	71.17%
100	73.80%	71.55%	72.58%	69.55%	71.14%
125	72.33%	71.21%	73.05%	71.61%	70.27%
150	71.83%	72.49%	72.55%	71.74%	71.36%
175	72.43%	71.77%	71.27%	72.49%	70.80%
200	72.68%	71.24%	71.30%	72.14%	71.27%
UI	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	62.12%	60.68%	62.12%	55.17%	55.17%
50	61.53%	62.97%	61.53%	64.32%	64.32%
75	61.78%	65.00%	61.78%	62.63%	62.63%

Component	*Feature Selection Algorithm*				
100	59.41%	63.64%	59.41%	62.29%	62.29%
125	60.93%	64.41%	60.93%	62.12%	62.12%
150	60.34%	63.39%	60.34%	65.34%	65.34%
175	60.51%	65.25%	60.51%	63.81%	63.81%
200	62.03%	65.25%	62.03%	62.97%	62.97%
Debug	*Chi square*	*Info-gain*	*PCA*	*Gini Index*	*Correlation*
Terms	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*	*Accuracy*
25	70.30%	69.93%	69.37%	71.12%	65.85%
50	71.52%	75.01%	73.61%	74.88%	73.61%
75	74.71%	76.57%	74.08%	76.34%	75.97%
100	76.14%	75.08%	74.78%	75.54%	75.47%
125	76.00%	75.31%	74.21%	74.84%	76.10%
150	75.97%	75.41%	74.84%	75.47%	74.58%
175	75.54%	75.77%	75.64%	75.37%	75.94%
200	74.94%	74.94%	74.94%	74.94%	74.94%

The maximum result is achieved using approximately 100 numbers of terms in all the components and maximum result is 76% using 5-NN classifier.

6. CONCLUSION

There are large numbers of bugs that are reported in bug tracking system. Automation of predicting severity of bug reports can provide a great help. Different feature selection algorithms for selection of severity indicator terms are used. Machine learning algorithms are used for the prediction purpose. From the result following observations are made:

1. The result stabilizes after 100 severity indicator terms by using approximately all feature selection methods used in this experiment.
2. Chi square and correlation methods has comparatively yielded best indicator of severity than other feature selection methods.
3. Different mining algorithms are used for finding the list of precedence. From the observations, it is found that 5-NN outperform other classifier, then SVM, Naïve bayes multinomial and Naïve bayes. Thus their precedence list obtained is.

   5-NN>SVM> NBM>NB.

   In future, the proposed approach can be validated using database of bug reports of other open source software.

REFERENCES

[1] Hassan AE. The road ahead for mining software repositories. InFrontiers of Software Maintenance, 2008. FoSM 2008. 2008 Sep 28 (pp. 48-57). IEEE.

[2] Raymond E. The cathedral and the bazaar. Knowledge, Technology & Policy. 1999 Sep 1;12(3):23-49..

[3] "15 Most Popular Bug Tracking Software to Ease Your Defect Management Process", http://www.softwaretestinghelp.com/popular-bug-tracking-software/".

[4] Anvik J. Assisting bug report triage through recommendation.

[5] Čubranić D. Automatic bug triage using text categorization. InIn SEKE 2004: Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering 2004.

[6] Anvik J, Hiew L, Murphy GC. Who should fix this bug?. InProceedings of the 28th international conference on Software engineering 2006 May 28 (pp. 361-370). ACM.

[7] Anvik J. Assisting bug report triage through recommendation.

[8] Menzies T, Marcus A. Automated severity assessment of software defect reports. InSoftware Maintenance, 2008. ICSM 2008. IEEE International Conference on 2008 Sep 28 (pp. 346-355). IEEE.

[9] Antoniol G, Ayari K, Di Penta M, Khomh F, Guéhéneuc YG. Is it a bug or an enhancement?: a text-based approach to classify change requests. InProceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds 2008 Oct 27 (p. 23). ACM.

[10] Lamkanfi A, Demeyer S, Giger E, Goethals B. Predicting the severity of a reported bug. InMining Software Repositories (MSR), 2010 7th IEEE Working Conference on 2010 May 2 (pp. 1-10). IEEE.

[11] Lamkanfi A, Demeyer S, Soetens QD, Verdonck T. Comparing mining algorithms for predicting the severity of a reported bug. InSoftware Maintenance and Reengineering (CSMR), 2011 15th European Conference on 2011 Mar 1 (pp. 249-258). IEEE.

[12] Tian Y, Lo D, Sun C. Information retrieval based nearest neighbor classification for fine-grained bug severity prediction. InReverse Engineering (WCRE), 2012 19th Working Conference on 2012 Oct 15 (pp. 215-224). IEEE.

[13] Ihara A, Kamei Y, Monden A, Ohira M, Keung JW, Ubayashi N, Matsumoto K. An Investigation on Software Bug-Fix Prediction for Open Source Software Projects--A Case Study on the Eclipse Project. InSoftware Engineering Conference (APSEC), 2012 19th Asia-Pacific 2012 Dec 4 (Vol. 2, pp. 112-119). IEEE.

[14] Park J, Lee M, Kim J, Hwang S, Kim S. Costriage: A cost-aware triage algorithm for bug reporting systems. InProceedings of the National Conference on Artificial Intelligence 2011.

[15] Sharma M, Kumari M, Singh VB. Understanding the meaning of bug attributes and prediction models. InProceedings of the 5th IBM Collaborative Academia Research Exchange Workshop 2013 Oct 17 (p. 15). ACM.

[16] Zanetti MS, Scholtes I, Tessone CJ, Schweitzer F. Categorizing bugs with social networks: a case study on four open source software communities. InProceedings of the 2013 International Conference on Software Engineering 2013 May 18 (pp. 1032-1041). IEEE Press.

[17] Bhattacharya P, Ulanova L, Neamtiu I, Koduru SC. An empirical analysis of bug reports and bug fixing in open source android apps. InSoftware Maintenance and Reengineering (CSMR), 2013 17th European Conference on 2013 Mar 5 (pp. 133-143). IEEE.

[18] Alenezi M, Magel K, Banitaan S. Efficient bug triaging using text mining. Journal of Software. 2013 Jan 9;8(9):2185-90.

[19] Yang CZ, Chen KY, Kao WC, Yang CC. Improving severity prediction on software bug reports using quality indicators. InSoftware Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on 2014 Jun 27 (pp. 216-219). IEEE.

[20] Kanwal J, Maqbool O. Managing open bug repositories through bug report prioritization using SVMs. InProceedings of the International Conference on Open-Source Systems and Technologies, Lahore, Pakistan 2010 Dec.

[21] Kanwal J, Maqbool O. Bug prioritization to facilitate bug report triage. Journal of Computer Science and Technology. 2012 Mar 1;27(2):397-412.

[22] Sharma M, Bedi P, Singh VB. An empirical evaluation of cross project priority prediction. International Journal of System Assurance Engineering and Management. 2014 Dec 1;5(4):651-63.

[23] Gujral S, Sharma G, Sharma S. Classifying bug severity using dictionary based approach. InFuturistic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015 International Conference on 2015 Feb 25 (pp. 599-602). IEEE.

[24] Sharma G, Sharma S, Gujral S. A Novel Way of Assessing Software Bug Severity Using Dictionary of Critical Terms. Procedia Computer Science. 2015 Dec 31;70:632-9.

[25] Bugzilla: https://bugs.eclipse.org/bugs/ [Last Access-6/21/2015].

[26] Bettenburg N, Just S, Schröter A, Weiss C, Premraj R, Zimmermann T. What makes a good bug report?. InProceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering 2008 Nov 9 (pp. 308-318). ACM.

[27] Antoniol G, Ayari K, Di Penta M, Khomh F, Guéhéneuc YG. Is it a bug or an enhancement?: a text-based approach to classify change requests. InProceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds 2008 Oct 27 (p. 23). ACM.

[28] Lovins JB. Development of a stemming algorithm. Cambridge: MIT Information Processing Group, Electronic Systems Laboratory; 1968 Mar 11.

[29] Willett, Peter. "The Porter stemming algorithm: then and now." Program 40.3 (2006): 219-223.

[30] Paice CD. An evaluation method for stemming algorithms. InProceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval 1994 Aug 1 (pp. 42-50). Springer-Verlag New York, Inc..

[31] Frakes WB. Stemming Algorithms.

[32] Mayfield J, McNamee P. Single n-gram stemming. InProceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval 2003 Jul 28 (pp. 415-416). ACM.

[33] Jivani AG. A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl. 2011 Nov;2(6):1930-8.

[34] Povey D, Ghoshal A, Boulianne G, Burget L, Glembek O, Goel N, Hannemann M, Motlicek P, Qian Y, Schwarz P, Silovsky J. The Kaldi speech recognition toolkit. InIEEE 2011 workshop on automatic speech recognition and understanding 2011 (No. EPFL-CONF-192584). IEEE Signal Processing Society.

[35] Aizawa A. An information-theoretic perspective of tf–idf measures. Information Processing & Management. 2003 Jan 31;39(1):45-65.

[36] Rapid miner tool: https://rapidminer.com/.