Domain Specific Automated Triaging System for Bug Classification

Heena Singla*, Gitika Sharma and Sumit Sharma
Department of CSE, CU, Gharuan, Mohali - 140413, Punjab, India;
Singlaheena77@gmail.com, gitikasharma41@gmail.com, sumit_sharma@mailingaddress.org

Abstract
The objective of this paper is to analyze and identify domain specific priority classification of bug reports. Different classification algorithms namely-Linear Discriminant Analysis (LDA), Naive Bayes (NB) to predict the performance measures are used. The performance of classification algorithms are compared using bug report instances of two open source software (OSS) namely NetBeans and Eclipse is downloaded from Bugzilla bug repository. Principal Component Analysis (PCA) for feature selection, Particle Swam Optimization (PSO) for instance selection to reduce the dimension of the bug reports. The results are compared based on four performance measures i.e. Accuracy, Precision, Recall and Processing Time. Feature selection to Instance selection gives better results than Instance selection to Feature selection is the result influenced from experiment performed. This approach shows that linear Discriminant analysis performs better than Naïve Bayes classifier.

Keywords: Bug Tracking System, Bug Trigging, Feature Selection, Instance Selection, Supervised Algorithms Software Testing

1. Introduction
Software is distinct from other physical methods in which inputs are accepted and outputs are produced1. Here, changes in software are inevitable. Changes in software are commonly requested during the development of any vast software system. Most of the requests are related to the software corrective maintenance assignment and these are termed as bug reports2. Software costs are mainly spent on software maintenance. Most of the large software system use bug tracking systems to track the bugs detected at user site. The bug reports reported in these Bug Tracking System (BTS) are then manually assigned to respective developer for fixation3,4. These bug tracking systems usually maintain a database to collect and handle the huge amount of bug reports that are known as bug repositories.

As reported in literature review5-7, bug tracking systems track large number of bugs. Therefore, it becomes challenging task for triager to triage the bugs within the accessible time and resources8 and also to decide which bug should be resolved immediately. Figure 1. Shows the whole process of corrective maintenance.

*Author for correspondence
Priority of bugs can be used to recognize the bugs that need instant concentration. The user informs the bug and allocates its priority that "how essential the bug is". Many times priority cell is left empty. Furthermore, the level of field may not be correctly allocated by user because his estimation about the significance of a bug may be diverse from other user.

The works described in this research paper classify bug reports of specific domain. According to our knowledge, there has been no work done on classifying bug reports of one specific domain to create bug triaging system more proficient. In that case, System need to be automatic for classification of bugs based on priority. The huge amount of new bug reports given to triager to manage them, it would be helpful to expand text mining methods that help in triaging to allocate priorities to the newly coming bug reports automatically.

Many researchers have been applied text mining and machine learning to make bug triaging system automatic and for finding redundancy in bug reports but still at this time bug triaging process has scope for enhancements. So, an experiment is made in this paper to estimate the domain specific priority of bug reports automatically using text mining approach and machine learning algorithms.

The remaining paper is described as follows: Section 2 explains Literature Review. Work Done for priority classification is explained in Section 3. Results obtained in experiment are discussed in Section 4. Conclusion and future scope is discussed in section 5.

2. Literature Review

Few attempts have been made to obtain Domain specific priority of bugs reports using supervised machine learning algorithms. Literature Review briefly discussed as below:

D. Cubranic, G. C. Murphy discussed about the ML algorithms which assigns bug automatically. Author applied machine learning algorithm i.e. Naïve Bayes for finding similarity between the expertise of the project developers and the new bug record.

John Ak et al. presented an approach of semi automating the allocation of a bug report to a developer with the suitable knowledge to reduce the information and used a supervised ML algorithm that was applied to information in the bug repository.

Herraiz, Israel et al. analyzed the bug reports of Eclipse. It was concluded that these bug report has too many options for severity and priority fields.

Shiv kumar et al. explained the Machine Learning (ML) classifiers to predict the presence of a bug and proposed a feature selection approach appropriate for classification based on bug prediction.

J Arturo Olvera-López et al. explained instance selection is most important part to classify new instances and reduced training set. Many researchers had given different algorithms or techniques to solve the problem of instance selection.

Xuan et al. explained the problem of the developer prioritization which aims to rank the contributions of developers and explore two tasks to empirically investigate the performance of our model.

Banu DIRI et al. introduced the extraction and selection method for text classification. Seven different algorithms explained and test based on the classifiers.

Neelofar et al. presented an automation system to classify software bugs. Many authors were applied feature selection techniques and machine learning algorithms for classification and achieve 83 % accuracy.

Ahmed Lamkanfi et al. proposed the Eclipse and Mozilla Defect Tracking Dataset filtered to contain only genuine defects and designed to cover the whole bug triage life cycle and used these datasets to predict bug severity, bug fixing time and identifying the assignment components.

Yuan Tian et al. presented the multiple factors that potentially affect the priority level of bug reports i.e. temporal, textual, author, related-report, severity, and product. Many authors automate the approach using supervised learning that would suggest a priority level based on description available in bug reports.

Shadi et al. discussed that bug triage system based on machine learning experience from low estimation accuracy. Author gives solution by proposing TRAM (Triaging Approach using bug reports Metadata) to improve the accuracy of bug triage.

Jifeng Xuan, He Jiang combined the feature selection and instance techniques which helps to reduce of data set and enhance the quality of the bug data. To find out the order of reduction techniques. Researchers analyze the data reduction for bug triage.

Neetu Goyal, Naveen Aggarwal find the priority using classification techniques on clustered bug report.
Gitika Sharma\textsuperscript{21} proposed an method to create a dictionary of critical terms which specifying severity levels using supervised learning approach. Use Naive Bayes Multinomial (NBM) and K-Nearest Neighbor (KNN). K-nearest neighbor present better classification for bug severity.

From the literature review, it is concluded that software bug effects the quality of software. Bug classification is very important part in bugs fixation process\textsuperscript{8}. However bug classification is manual process. A lot of research has been performed to automate bug classification but no attempt has been made to classify bug reports of one specific domain on the basis of priority\textsuperscript{21}. This paper gives the solution for priority classification of bug reports using text mining and machine learning algorithms.

3. Work Done

Work Done is important part of research because in this section whole experimental work is discussed. Work done section includes dataset acquisition, Pre-processing steps, Feature Selection and Instance Selection Techniques, Training and Testing. The entire process of priority prediction is proposed in this paper describe in Figure 2.

3.1 Dataset Acquisition

Eclipse and NetBeans bug reports are acquired from bugzilla\textsuperscript{22}. These are integrated development environment (IDE) that are used globally for developing software. These bug reports represents the bugs that are encountered by user or developers themselves while operating it. The another reason to take bug reports of this domain is that users of these IDE are developers themselves so bug reports reported should be of high quality.

Bug reports has too many fields but bug id, Summary and priority cells of bug reports are extracted for purpose of classification. Priority levels in bug reports has five levels P1,P2,P3,P4 and P5 where P1 represents highest priority and P5 lowest Priority. But in\textsuperscript{9} it is observed that priority levels can be reduced from five levels to three levels i.e. Higher level, Medium level and lower level. Priority level P1 and P2 can be considered as higher level, P4 and P5 as lower level and P3 as medium level of priority. This research considers bug reports of two same components of Eclipse and NetBeans i.e. Debugger and ANT. The bug reports of same component are combined for the purpose of making it reports of one domain to validate our approach. Combined bug reports count of Eclipse and NetBeans is given as:

<table>
<thead>
<tr>
<th>Components</th>
<th>Higher Level</th>
<th>Medium level</th>
<th>Lower Level</th>
<th>Total Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugger</td>
<td>1195</td>
<td>2896</td>
<td>199</td>
<td>4290</td>
</tr>
<tr>
<td>Ant</td>
<td>745</td>
<td>4631</td>
<td>235</td>
<td>5611</td>
</tr>
</tbody>
</table>

3.2 Pre-Processing Steps

Summary as an attribute is used for pre-processing steps and it is also included for prediction of priority levels\textsuperscript{9}. Detail description of bug report does not give better results than summary of bug reports\textsuperscript{23}. Pre-processing steps includes tokenization, stop word removal and stemming.

- Tokenization: Tokenization separate the entire document into tokens to expand the words in document. For Example: What is Software. Then tokenization process divides into tokens like “What”, “is” and “Software”.
- Stop word removal: The Stop words (like is, am, has etc.) are those words which are very common and rarely contain any important information. Thus, it decrease textual data and increase system performance. For example, if a search engine is searching query “what is Machine learning” then search engine will find a lot of irrelevant page containing terms “what” and “and”, however “Machine” and “learning” terms would help to find relevant web pages of Machine learning.
- Stemming: Stemming is used to reduce the words to their root words e.g. words like “fishing”, “fished” and “fisher” has its root word “Fish”.

3.3 Term Document Matrix

Term Document Matrix (TDM) is generated after completing pre-processing steps. Every column represents the terms occurring in documents and row represents every bug report. The rows and columns of matrix are loaded with binary values. If term is not present in the specific bug report then rows and columns are filled with 0 otherwise 1.
3.4 Feature Selection using PCA

Feature selection approach is used to recover the most revealing terms from amount of matrix. This study used Principal Component Analysis (PCA) algorithm to compute and study the Eigenvectors to find the characteristics values and then to direct each data with its principal components or Eigenvectors.

3.5 Instance selection using Particle Swarm Optimization

Instance selection is a data reduction technique. Main focus of instance selection is to remove some duplicate instances from a given dataset. It deals with selection of instances to reduce the size of the matrix and will ease in processing to deal with the further proceedings input. Particle Swarm Optimization (PSO) is instance based method which is used to reduces the instances of the bug reports.

3.6 Training and Testing

Linear Discriminant Analysis (LDA) and Naive Bayes (NB) are used for classification purpose. Training and testing is done with 10 cross validation. The dataset is divided into 10 parts, testing dataset assumed a single subset and training datasets are taken remaining nine subsets. The process of cross validation is repeated 9 times considering each of the subsets as training dataset. Then ten results find after ten iterations and result is analyzed. Performance is evaluated with the help of Accuracy, Precision, Recall and Processing time.

4. Results and Discussion

In order to analyze the performance of our approach, bug report instance of two Open Source Software (OSS) namely NetBeans and Eclipse is downloaded from Bugzilla bug repository. Bug reports same component of these OSS are combined and considered as bug reports.
of software of one domain. These bug reports have five classes P1, P2, P3, P4 and P5. The bug reports are classified as high, medium and low. The bug reports of P1 and P2 priority are considered as high priority bug reports, P3 priority bug reports are considered as medium bug reports and P4 and P5 priority bug reports are considered as low priority bug reports. Experiment is performed on these bug reports and following results are obtained.

**Feature selection to Instance selection for LDA**

**Table 2.** Feature selection to instance selection for LDA

<table>
<thead>
<tr>
<th>Components</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugger</td>
<td>97.6151</td>
<td>76.914</td>
<td>87.522</td>
</tr>
<tr>
<td>ANT</td>
<td>80.7008</td>
<td>67.78</td>
<td>93.51</td>
</tr>
</tbody>
</table>

- Table 2 shows Accuracy varies from 80.7008% to 97.6151%
- While the precision of debugger component in case of FS to IS Ranges from 67.78% to 76.914%
- Recall achieves 87.522% to 93.51%

**Feature selection to Instance selection for Naive Bayes**

**Table 3.** Feature selection to Instance selection for Naive Bayes

<table>
<thead>
<tr>
<th>Components</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugger</td>
<td>91.595</td>
<td>62.496</td>
<td>81.658</td>
</tr>
<tr>
<td>ANT</td>
<td>62.294</td>
<td>33.196</td>
<td>33.196</td>
</tr>
</tbody>
</table>

- Table 3 show that feature selection to instance selection using Naive Bayes
- The accuracy lies between the range 62.29% to 91595%
- Precision lies between 33.196% to 62.496
- Recall achieves 33.196% to 81.658%

**Processing time FS to IS for LDA and NB**

**Table 4.** Processing time FS to IS for LDA and NB

<table>
<thead>
<tr>
<th>Components</th>
<th>Processing time(s)(LDA)</th>
<th>Processing time(s) (NB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugger</td>
<td>1.081</td>
<td>30.3082</td>
</tr>
<tr>
<td>ANT</td>
<td>35.9141</td>
<td>78.6243</td>
</tr>
</tbody>
</table>

- As Shown in Table 4.Processing time of FS to IS for LDA and NB shown in table 4, it achieves 1.081s to 35.9141s for LDA and 30.3082s to 78.6243s for NB.

**Instance selection to Feature selection for LDA**

**Table 5.** Instance selection to Feature selection for LDA

<table>
<thead>
<tr>
<th>Components</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugger</td>
<td>53.4639</td>
<td>18.257</td>
<td>03.332</td>
</tr>
<tr>
<td>ANT</td>
<td>26.6096</td>
<td>18.257</td>
<td>33.332</td>
</tr>
</tbody>
</table>

- Accuracy varies from 53.4639% to 26.6096%
- While the precision rates remain same is 18.257
- Recall achieves 03.332% to 33.332%. As shown in Table 5.

**Instance selection to Feature selection for Naive Bayes**

**Table 6.** Instance selection to Feature selection for Naive Bayes

<table>
<thead>
<tr>
<th>Components</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugger</td>
<td>81.7427</td>
<td>18.257</td>
<td>03.332</td>
</tr>
<tr>
<td>ANT</td>
<td>20.4357</td>
<td>18.257</td>
<td>03.332</td>
</tr>
</tbody>
</table>

- The accuracy lies between the range 20.4357 % to 81.7427% . As shown in Table 6.
- Precision rates for debugger and ant component was same is 18.257
- Recall achieves 03.3327% to 03.3332%

**Processing time IS to FS for LDA and NB**

**Table 7.** Processing time IS to FS for LDA and NB

<table>
<thead>
<tr>
<th>Components</th>
<th>Processing time(s)(LDA)</th>
<th>Processing time(s) (NB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugger</td>
<td>0.46484</td>
<td>35.6947</td>
</tr>
<tr>
<td>ANT</td>
<td>0.40321</td>
<td>0.18257</td>
</tr>
</tbody>
</table>

- Processing time of IS to FS for LDA and NB shown in Table 7.
- Debugger component consumes 1.081s to 35.9141s for LDA
- ANT component consumes 30.3082s to 78.6243s for NB

**LDA comparison of Debugger component**

**Table 8.** LDA comparison of Debugger component

<table>
<thead>
<tr>
<th>Components</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS to IS</td>
<td>97</td>
<td>76.914</td>
<td>87.522</td>
</tr>
<tr>
<td>IS to FS</td>
<td>53.4639</td>
<td>18.257</td>
<td>33.332</td>
</tr>
</tbody>
</table>

- The comparisons of classifiers between reduction order's. As shown in Table 8.
• FS to IS gives best results than IS to FS
  • Accuracy of LDA changes 53.4639% to 97%
  • Precision rate lies between 18.257% to 76.914%
  • Recall achieves 33.327% to 87.522%

Naive Bayes comparison of Debugger component

Table 9. Naive Bayes comparison of Debugger component

<table>
<thead>
<tr>
<th>Components</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS to IS</td>
<td>91.595</td>
<td>62.496</td>
<td>81.658</td>
</tr>
<tr>
<td>IS to FS</td>
<td>81.742</td>
<td>18.257</td>
<td>33.327</td>
</tr>
</tbody>
</table>

• Table 9. shows that the accuracy range between 81.742% to 91.595%
  • Precision rate achieves 18.257% to 81.658%
  • Recall rate is 33.327% to 81.658%.

Processing Time Debugger component for LDA and NB

Table 10. Processing Time debugger component for LDA and Naïve Bayes

<table>
<thead>
<tr>
<th>Components</th>
<th>Processing time(s) (LDA)</th>
<th>Processing time(s) (NB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS to IS</td>
<td>0.37687</td>
<td>22.4163</td>
</tr>
<tr>
<td>IS to FS</td>
<td>0.46646</td>
<td>27.1548</td>
</tr>
</tbody>
</table>

• In case of FS to IS processing time for LDA lies between 0.37687s to 0.46646s
  • In case of IS to FS processing time for NB lies between 22.4163s to 27.1548s
  • Table 10. shows Processing Time Debugger component for LDA and NB

LDA comparison of ANT component

Table 11. LDA comparison of ANT component

<table>
<thead>
<tr>
<th>Components</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS to IS</td>
<td>80.7008</td>
<td>67.78</td>
<td>93.51</td>
</tr>
<tr>
<td>IS to FS</td>
<td>26.6096</td>
<td>18.25</td>
<td>03.33</td>
</tr>
</tbody>
</table>

• Accuracy lies between the range 26.6096% to 80.7008%
  • While precision lies 18.25% to 67.78%

• Recall rate is 03.33% to 93.51%. As shown in Table 11.

Naive Bayes comparison of ANT component

Table 12. Naive Bayes comparison of ANT component

<table>
<thead>
<tr>
<th>Components</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS to IS</td>
<td>62.2948</td>
<td>33.196</td>
<td>88.233</td>
</tr>
<tr>
<td>IS to FS</td>
<td>20.4357</td>
<td>18.257</td>
<td>03.333</td>
</tr>
</tbody>
</table>

• Accuracy lies between the range 20.4257% to 62.2948%
  • While precision lies 18.25% to 33.196%
  • Recall rate is 03.33% to 88.233%. As shown in Table 12.

Processing time ANT component for LDA and NB

Table 13. Processing time ANT component for LDA and NB

<table>
<thead>
<tr>
<th>Components</th>
<th>Processing time(s) (LDA)</th>
<th>Processing time(s) (NB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS to IS</td>
<td>0.38623</td>
<td>18.6477</td>
</tr>
<tr>
<td>IS to FS</td>
<td>0.42559</td>
<td>30.5124</td>
</tr>
</tbody>
</table>

• In case of FS to IS processing time for LDA lies between 0.38623s to 0.42559s
  • In case of IS to FS processing time for NB lies between 18.6477s to 30.5124s as shown in Table 13.

Comparison of different approaches

From the experimental results, it is analyzed that FS to IS order enhances the performance of LDA. Furthermore, the data reduction ability of FS to IS have better results than IS to FS. As shown in Figure 3. and Figure 4.
The feature extraction with instance selection is a better approach than instance selection with feature extraction because there is less variance from the mean value, which produces high accuracy of retrieving data during classification process on the basis of training data. This produces less standard deviation. When input consists of a large data set to be processed, it will build feature values in terms of feature vector to reduce redundancy for efficient interpretations and transform it to a reduced data set of values so that the desired task will be performed in an efficient manner on the reduced dataset instead of using the whole initial data. As shown in Figure 5. and Figure 6.

In terms of precision, recall, accuracy and computation time, it can be observed that LDA performs better than Naïve Bayes (NB) as shown in Figure 7. and Figure 8. As Naïve Bayes is having high independence assumptions of features which depend on predictable criteria as a result of which the classification is not up to the mark as compared to LDA. As LDA is a regression-based model which outperforms the less variance with less standard deviation which results in the characterization of two or more classes of events. As shown in Figure 9. and Figure 10.
10. It uses the linear combination of feature values which provides best explanation of the data.

5. Conclusion and Future Scope

This paper attempts to make domain specific automated triaging systems that classify bug reports based on priority level. For this approach bug reports of common components of two widely used IDEs are used. Data reduction techniques with two machine learning algorithms LDA and NB are applied on bug reports and performance is evaluated. It is observed that the performance of LDA is better than NB and also the reduction order Feature to Instance selection gives better results in terms of Accuracy, Precision, Recall and processing time.

In future this domain specific approach can be validated on other bug reports of one domain. Further a comparison can be conducted among other machine learning algorithms.

6. Reference


