BUG REPORT CLASSIFICATION

Final Report

DECEMBER 19, 2017

BSSE0623
IIT, DU

Submitted by

Md. Habibur Rahman

BSSE0623

Supervised by

Shah Mostafa Khaled

Assistant Professor

ITT, DU

Submission Date

19/12/2017

ABSTRACT

Now-a-days there are lots of vast and complex software are developed. For maintenance of these software continuously we need to resolve different bugs and add some feature which are issued by users. For these vast and complex software there are lots of issues are reported by users and to resolve these issues we need to classify them. But classifying such huge number of issues are so difficult and time consuming. For this we need an automated tool which will classify these issues. But which machine learning approach will give us more accurate result. So, here we have done such comparative analysis among different approaches. For this comparative analysis we have used same dataset for all approaches and compare them based their accuracy, precision, recall and f-measure.

Acknowledgement

I would like to elicit my gratitude to my supervisor Shah Mostafa Khaled for his motivation and guidance during the thesis completion. All of his suggestions and directions were so helpful and he was very careful and imperturbable to me.

Contents

1 Introduction

A bug is a defect in a system which indicates the abnormal behaviour of a system. Software bugs are managed and tracked using different bug tracking tools like Bugzilla, JIRA etc. Software bug repositories contain the software bug information reported by the users. When a bug is reported the QA manager manually inspects the bug report and then he takes necessary steps like assigning to the developers, predict the severity of the bug, classify the bug report as bug or enhancement etc.

All bug reports reported by the users are not actually bug. A bug report can be a feature request, a change in documentation etc. which are called non-bug. The QA manager manually checks a bug report whether it is bug or non-bug. The QA manager classifies the bug report based on the bug report attributes. The attributes of bug report are title, description, assigned to, date, comment etc. Everyday a huge number of bugs are reported in bug repositories. For this, classifying huge number of bug reports by the QA manager is time consuming. An automated approach in classifying bug reports as bug and non-bug can reduce the time and effort of QA manager. But the existing bug tracking tools do not support this automation approach.

To classify automatically we can use different machine learning approach. To know which approach is good for this classification we need a comparative analysis. Here we make an attempt to do such analysis. For this analysis here we have used similarity histogram based clustering, naïve bayes, logistic regression, decision tree and self-learning.

2 Background Study

Software bug repositories are valuable assets for software maintenance activity. Users reports bugs to these repositories. Users of these repositories are usually non-technical and cannot assign correct class to these bugs and sometimes they report some bugs which are not actually bug. For these QA managers need to classify the reports. But these kind of activities are very tedious and time consuming. Due to these automatic bug classification is very important in software industry.

2.1 Bug Classification

A software bug means an error in a system for which it shows unexpected behaviour. Bug classification means classify the software bugs into different categories. We can classify as bug, non-bug or different types of bugs like logical bug, UI bug, security bug etc. There are different software bug repositories where the bug are reported. The QA manager then classify the bugs and assign those to corresponding developers.

2.2 Bug Repository

To report the bugs there exist several open source bug tracking system also known as bug repository like as Bugzilla, Jira, Mozilla, Eclipse etc. Bugs are reported in these repositories. To report a bug we need to provide some information like title, description, severity, priority etc. and a unique identification number is assigned to this report.

2.3 Stop Words Elimination and Stemming

For text classification stop words elimination is very important. Stop words are that words which has no significant meaning but it increase complexity in classification. There is no universal stop words list it depends on application. Stemming is the process by which we convert a word to its original form like converting the words argue, argued, argues, arguing and argus to its root form argu.

2.4 Machine Learning Techniques

A classifier is a function that assign a label from finite set of classes to observations. There are three types of machine learning techniques are available for classification-

- Unsupervised learning
- Supervised learning
- Reinforcement learning

Unsupervised learning like clustering classify the data based on some fitness or cost function like similarity or distance. Unsupervised learnings are most popular because there is no need to label the data, but it is very hard in some context to interpret the resulting classification and linking the classification with characteristics of data.

There are several clustering techniques like-

- K-Means
- Hierarchical Clustering
- Gaussian Mixture Model
- Hidden Markob Model
- Self-Organizing Map

Supervised learning means deducing a function from labelled training data. Each training data is consists of input vector and supervisory signal. The supervised learning algorithms analyse the training data and deduce a function which is used to know the label of new data. This function is built by maximizing the gain or minimizing the cost.Some supervised learning algorithms are –

- Naïve Bayes
- Decision Tree
- Support Vector Machine
- Linear Regression etc.

2.5 Cosine Similarity

Cosine similarity is a measure of similarity between two vectors of an inner product space that measure the cosine of the angle between them. The cosine of zero degree is one and it is less than one for any other angle.

$$COS\Theta = a.b/\sqrt{a^2}.\sqrt{b^2} \quad \ldots\ldots\ldots\ldots 2.1$$

Steps of calculating cosine similarity between two texts:

Step1: Identify all distinct words in two texts

Step 2: Identify all frequency of occurrences of these words in both text and treat it as vector

Step 3: Apply cosine function

Suppose we have two texts

Text1: Julie loves me more than linda loves me

Text2: Jana likes me more than Julie loves me

For the given texts, we can use the following vector representation

Distinct words	Frequency in text1	Frequency in text2
Julie	1	1
Loves	2	1
Me	2	2
More	1	1
Than	1	1
Linda	1	0
Jana	0	1
Likes	0	1

VectorA = [1, 2, 2, 1, 1, 1, 0, 0]

Vector B = [1, 1, 2, 1, 1, 0, 1, 1]

So, a.b = 9 and √(a^2) = 12, √(b^2) = 10

Cosine similarity = 0.822

2.6 A Similarity Histogram based Sentence Clustering

This clustering approach is an incremental dynamic method of building the sentence clusters [6]. In incremental clustering, every objects are processed one at a time and every objects are incrementally assigned to their respective clusters while they progress. The main concept for the similarity histogram-based clustering formula is to keep each cluster as coherent as possible and a degree of coherency in a cluster at any time is observed with a Cluster Similarity Histogram. Cluster similarity histogram is a concise representation of the set of pair-wise sentence similarities distribution in a cluster. A histogram consists of a number of bins that correspond to fixed similarity value intervals. Each bin height represents the count of pair-wise sentence similarities in the corresponding interval.

A perfect cluster would have a histogram, where all pair-wise similarities are of maximum value and the histogram would have the right-most bin representing all similarities. On the other hand, a loose cluster would have histogram where all pair-wise similarities are minimum and the similarities would tend to be counted in the lower bins.

To prevent selection of the redundant sentences into the summary, we should be careful to keep each cluster as coherent as possible. In other words, the objective would be to maximize the number of similarities in the high similarity intervals. To achieve this goal in an incremental fashion, we should judge the effect of adding a new sentence to a certain cluster. If the inclusion of this sentence is going to degrade the distribution of the similarities in the clusters very much, it should not be added, otherwise it is added. But assignment of sentences to clusters based on similarity distribution enhancement may create problem with the perfect clusters. The sentence may be rejected by the perfect cluster even if it has high similarity to most of the sentences in the cluster. So, the quality of a similarity histogram representing cluster cohesiveness is judged by calculating the ratio of the count of similarities above a certain similarity threshold to the total count of similarities. The higher this ratio, the more coherent the cluster is.

If n be the number of the documents in a cluster, the number of pair-wise similarities in the cluster is n(n+1)/2. Let S = {sim_i, i = 1,..., m} be the set of pair-wise sentence similarities in a cluster, where m=n(n+1)/2. The histogram of the similarities in the cluster is given by

H={hi , i=1, ..., n_b}

hi = count(sim_k) sim_{li}<=sim_k< = sim_{ui},

where,

n_b: the number of bins in a histogram

hi : the count of sentence similarities in bin i

sim_{li}: the lower similarity bound of bin i

sim_{ui}: the upper similarity bound of bin i

The histogram ratio of a cluster is calculated using the following formula:

$$HR = \sum_{i=T}^{nb} hi \,/\, \sum_{j=1}^{nb} hj \quad \text{..... 2.2}$$

$$T = S_T * n_b \text{ ..... 2.3}$$

S_T = the similarity threshold

T = bin number corresponding to the similarity threshold

Algorithm for clustering:

1) Say, Clist is a cluster list which is initially empty

2) Convert all the input documents in a collection of sentences, Slist. Each sentence S in Slist is indexed by the document number and the sentence number

3) For each sentence S in Slist do for each cluster c in Clist do

3.1 Store the histogram ratio of the cluster c to a variable before adding s to c, that is,

$HR_{old} = HR_c$

3.2 Simulate adding s to c to check whether addition of s to c would severely degrade or improve the histogram ratio (coherence) of c. Let the simulated histogram ratio be HRs

3.3 If ($HR_s \geq HR_{old}$) or (($HR_s \geq HR_L$)and ($HR_{old} - HR_s < eps$))

then add s to c and exit from the inner loop to avoid any chance of assigning the same sentence to more than one cluster.

3.4 If s is not added to any cluster, then create a new cluster c, add s to c and add c to Clist.

end for (inner loop)

end for (outer loop)

2.7 Naïve Bayes Classifier

Naïve Bayes is a simple supervised machine learning algorithm. Mainly it is a probability based classification algorithm. In Naïve Bayes algorithm it is assumed that all features independent and they contribute to the response independently. It performs better when the dataset is very large. This algorithm is very easy to build and the probability of a class is calculated by the following formula:

$$P(c \mid x) = (P(x \mid c).P(c))/(P(x)) \ldots\ldots\ldots\ldots 2.4$$

Here,

c and x are response class and predictors accordingly

P(c | x) is posterior probability

P(x | c) is likelihood

P(c) is class prior probability

P(x) is predictor prior probability

After calculating the posterior probability, we select the class based on highest posterior probability.

2.8 Logistic Regression

We need to use logistic regression when our response variable is categorical such as bug, non-bug. Suppose we want to classify the bug reports as bug, non-bug. Here our outcomes are categorical.

$$Y = \begin{cases} 0 \; if \; bug \\ 1 \; if \; non-bug \end{cases} \quad \ldots\ldots\ldots 2.5$$

Here, use of linear regression will not be suitable because linear regression gives us a continuous value which is negative infinity to positive infinity but our response is only 0 and 1. But linear regression and logistic regression are not so much different. The tricky point of logistic regression is it converts the linear regression output range (- ∞, + ∞) to (0, 1) by using following formula:

Suppose, our linear regression formula is:

$$p(X) = \beta 0 + \beta 1 X \ldots\ldots\ldots 2.6$$

Now logistic regression formula will be:

$$p(X) = \frac{e^{\beta 0+\beta 1X}}{1+ e^{\beta 0+\beta 1X}} \quad \ldots\ldots\ldots 2.7$$

2.9 Decision Tree

Decision tree is a tree based model which can be applied on both regression and classification model. As our problem is classification problem, here we discuss about classification model. In decision tree main challenge is to set the decision node. Here we select that feature as decision node which feature gives us best result from the training data and we split the data based on that feature. Then again we decide now which feature gives us best result and will continue until we get only the class. To take this decision we can use gini index or cross entropy.

Gini index:

$$G = \sum_{k=1}^{K} Pk(1 - Pk) \quad \ldots\ldots\ldots 2.8$$

Entropy:

$$D = \sum_{k=1}^{K} Pk \log Pk \quad \ldots\ldots\ldots\ldots 2.9$$

Here, Pk estimated probability of that class

3 Literature Review

Classification of software bugs using bug attribute similarity (CLUBAS) is proposed by N. K. Nagwaniet. al. [1]. The proposed work can be divided into five steps- data pre-processing, text clustering, frequent term calculations, taxonomic terms mapping techniques and performance evaluation. In the first step data are pre-processed by eliminating stop words and applying stemming over the textual bug attributes. In the second step text clusters are created using textual similarity between the attributes summary and description for each pair of bugs. In the third step cluster labels are generated calculating the frequent and meaningful terms from each cluster text data and assign them to that cluster. In the fourth step cluster labels are mapped against the bug taxonomic terms to identify appropriate categories of the clusters. In the fifth step performance is evaluated by calculating accuracy, precision, recall and F-measure.

The algorithm CLUBAS shows stability and performs better than Classification using Clustering (CC), Support Vector Machine (SVM) and J48 when this algorithm is implemented on the Android bug repository. It maintains the F-measure value more than 0.9 for each experiment using different number of samples. When the precision, recall and F-measures are important CLUBAS gives the better and stable results irrespective of number of samples and software bug repositories. But both Naïve Bayes (NB) and Naïve Bayes Multinomial (NBM) performs better in terms of accuracy than CLUBAS.

To improve the performance of CLUBAS advance text pre-processing techniques can be implemented to optimize the clustering and classification work and also modern text clustering and classification can be implemented.

Recent approaches in automatically classifying bug reports as bug and non-bug are based on text mining. Antoniol et al. [2] have investigated the automatic classification of bug reports by utilizing conventional text mining techniques, which demonstrated the feasibility. They have used title and description of bug reports. They have collected about 1800 bug reports from Mozilla, Eclipse and Jboss. The approach consists of three pipeline phases. First, they have manually classified the bug reports as bug or non-bug. Second, the classifier is trained by the labelled data. Third, they have predicted the bug reports of test data using the trained classifier. In this paper they use Alternating Decision Tree (ADTree), Naïve Bayes (NB) and linear logistic regression classifier and evaluate their performance by using 10-fold cross validation.

Here, for logistic regression and ADTree, the precision of bug and the recall of non-bug are increased when the number of features are increased though decreased the recall for bug, for Mozilla repository. But NB exhibits limited improvement with increasing the number of features. For eclipse the performance of three classifiers is similar though the improvement is very limited with the increasing of the number of features. For JBoss the three classifiers show poor performance when the number of features are low like 20. For all repository logistic regression performs better than other classifiers.

In the following paper they classified the bug reports only as bug and non-bug. Here, further classification can be possible like as logical bug, security bug, UI related bug etc. and the performance of the classifiers are not so high. So, there are some opportunity to increase the performance.

To assigning the right bug to right developer Muhammad [3] proposed an automated approach for software bug classification. Their work can be split into three major steps pre-processing, feature selection and classification. In pre-processing step at they eliminate the stop words and punctuations. After that they use porter stemming algorithm for stemming the vocabulary. In feature selection step they use Chi-Square and Term Frequency Inverse Document Frequency. And Naive Bayes text classifier is used for bug classification.

Using the following approach they get maximum 86% accuracy. Highest accuracy is obtained when the training to testing ratio is 1:11. The following approach gives the better result than Support Vector Machine (SVM) when the training data is small. From the processing time point of view it performs better than SVM. But when the training dataset is large SVM performs better. Here the performance can be increased by using other feature selection algorithm and the synonym dictionary can be used to improve the performance.

Feature extraction and comparison of event model using Naïve Bayes approach for bug classification is proposed by S J Dommati et al [4]. They give priority on feature extraction, noise reduction and classification of network bugs. They distribute their work as feature extraction and pre-processing, probabilistic framework for classification and performance measure. The goal of a bug feature extractor is to automatically extract features from bug information in bug repository. They consider title, description and crash file attachment as valuable attributes. In this paper they use Information Gain Measure as feature selection. They use Bernoulli and Multinomial Naïve Bayes for classification. Their assumption was there is strong independence of features. In the Bernoulli, a bug is represented as a binary vector over the space of features. On the other hand the Multinomial captures feature frequency information in bugs. They get 60% and 78% accuracy for Bernoulli and multinomial respectively. More bug specific feature selection and better classification can be used for better performance.

An empirical study on bug characteristics is proposed by Zhenmin Li [5]. They assert that a deep understanding on bug characteristics is very important for detecting and recovering from software failures. Their study has discovered several new interesting characteristics: (1) memory-related bugs have decreased because quite a few effective detection tools became available recently; (2) surprisingly, some simple memory-related bugs such as NULL pointer dereferences that should have been detected by existing tools in development are still a major component, which indicates that the tools have not been used with their full capacity; (3) semantic bugs are the dominant root causes, as they are application specific and difficult to fix, which suggests that more efforts should be put into detecting and fixing them; (4) security bugs are increasing, and the majority of them cause severe impacts. They use summary, status, description, severity, assignee, reporter and discussion comment bug attributes for their analysis. To produce classification models for

different bug categories they use manually labelled bugs as training set. They use Naive Bayes, Winnow, Perceptron and Support Vector Machine as classifier.

Summarization of multiple text documents based on sentence clustering is proposed by K Sarkar [6].Their work is split into three main steps clustering sentences, cluster ordering and selection of representative sentences. In the clustering step they use uni-gram matching based similarity measure and similarity histogram based sentence clustering.

J Xuan et al. [7] proposed a semi supervised text classification approach. Using this approach some deficiency can be avoided like labelling of bug reports in existing supervised approach. Their approach combines Naïve Bayes classifier and expectation maximization to take advantage of both labelled and unlabelled bug reports. They demonstrate two basic phases in their approach. In the first phase to train a classifier with labelled bug reports. In their approach they use Naïve Bayes classifier for the following reasons:

- Naïve Bayes is a probability classification which shows the better performance on text form of bug reports.
- Sorting them with probability is a common method for providing recommendation.
- It is easy to extend expectation maximization with a probability weight based on Naïve Bayes.

The second phase is expectation maximization with both labelled and unlabelled bug reports. Evaluation and labelling the bug reports on unlabelled subset is the expectation step and the maximization step is to rebuild the classifier with the label of all bug reports. Until the performance of classifier does not improve, the iterations of building classifiers is repeated.

4 Methodology

Methodology of different machine learning approaches to classify the bug reports are described in this chapter. Approaches are:

- Similarity Histogram based clustering (Unsupervised)
- Naïve Bayes (Supervised)
- Logistic Regression (Supervised)
- Decision Tree (Supervised)
- Self-Learning (Semi-Supervised)

4.1 Unsupervised Approach

In unsupervised approach similarity histogram based clustering is used. It is a text clustering approach using cosine similarity. Steps of this approach is given below:

- At first stop words are eliminated here. Stop words are eliminated because those have not significant impact for text classification.
- After that stemming is performed over the text data to unify the terms present in the document.
- Then similarity histogram based clustering algorithm is performed. The key concept of this algorithm is a new report is assigned to that cluster if the histogram ratio of that cluster is increased after the incorporating that report otherwise a new cluster will be created.
- After clustering cluster labelling is performed using frequent term calculation.

4.2 Supervised Approach

In supervised approach Naïve Bayes, Logistic Regression, Decision Tree are used. In this approach Chi Squire Test is used for feature selection. Following steps are used here:

- Stop words elimination and stemming is performed at first.
- After that feature selection algorithm is performed to identify the significant feature.
- Then data are prepared using the selected features.
- After preparing data Naïve Bayes, Logistic Regression, Decision Tree algorithms are performed individually.

4.3 Semi-Supervised Approach

Self-learning algorithm is used for semi-supervised approach. This approach also use Naïve Bayes algorithm. Steps are given below:

1. At first stop words elimination and stemming are performed to prepare the data.
2. Then data are divided into two parts. First part is labelled by manually.
3. Using labelled data classifier is trained.
4. A small chunk of unlabelled data are labelled using the trained classifier
5. Then classifier is trained again after incorporating new labelled data
6. Step 4 and 5 are performed until labelling the whole unlabelled data

5 Result Analysis

The objective of this chapter is to evaluate the performance of different approaches (Supervised, Semi-Supervised and Unsupervised) based on experimentation using same dataset on implemented tool. In this chapter we also analysis that how much the threshold of feature selection affect the result.

For result analysis and comparison we have used the same dataset which is taken from following paper, a dataset of high impact bugs by Masao Ohira [8]. The data is taken from four different projects named "The Apache Ambari project" [9], "The Apache Camel project" [10], "The Apache Derby project" [11] and "The Apache Wicket project" [12]. One thousand report has taken from each project. Following information are exist in the dataset:

- issue_id
- type
- status
- resolution
- component
- priority
- reporter
- created
- assigned
- assignee
- resolved
- time_resolved
- time_fixed
- summary
- description
- affected version
- fixed version
- votes
- watches
- description_words
- assignee_count
- comment_count
- commenter_count
- commit_count
- file_count
- files

From the following information we have used “type” and “summary” for our analysis. We also could use “description” for our analysis but we have used summary because it consist more precise information for classification.

5.1 Performance of Each Approaches

In this section we discuss the result of each approach. We also discuss the advantages and disadvantages of each approaches.

5.1.1 Unsupervised Approach

In unsupervised approach we use similarity histogram based clustering. The accuracy of this approach is around 64%. The precision of this approach is around 71% which means that 71% bugs are actually bug which it identifies as bug, on the other hand we also can say that 29% bugs are not actually bug. The recall of this approach is around 82% which means that among all bugs 82% bugs are classified as bug, we also can say that 18% bugs are classified as non-bug. F-Measure of this approach is 0.77.

Table 1: Result of similarity histogram based clustering

Result Name	Value
Accuracy	64%
Precision	71%
Recall	82%
F-Measure	0.76

The main advantage of this approach is it does not need to classify data manually as it cluster the data using the cosine similarity. The main drawback of this approach is its performance will degrade severely if the bug report contain so much noise. Text should be context specific and unnecessary information should avoid. Spelling mistake should be removed. Error location should not be mentioned because those do not have significant impact on classification but increase the scale of the data

5.1.2 Supervised Approach

In supervised learning we have used three approach Naïve Bayes, Logistic Regression and Decision Tree. For feature selection we have used chi-squire test. We have also run the following algorithm for different number of feature.

5.1.2.1 Naïve Bayes

Accuracy of this approach is 59% for 3000 features, 58% for 200 features, 57% for 100 features and 56% for 50 features. Precision of this approach is 56% for 3000 features, 57% for 200 features, 57% for 100 features and 52% for 50 features. Recall of this approach is 83% for 3000 features, 81% for 200 features, 80% for 100 features and 82% for 50 features. F-Measure of this approach is 0.67 for 3000 features, 0.67 for 200 features, 0.66 for 100 features and 0.64 for 50 features.

Table 2: Result of Naive Bayes Approach

Result Name	Number of Feature (3000)	Number of Feature (200)	Number of Feature (100)	Number of Feature (50)
Accuracy	59%	58%	57%	56%
Precision	56%	57%	57%	52%
Recall	83%	81%	80%	82%
F-Measure	0.67	0.67	0.66	0.64

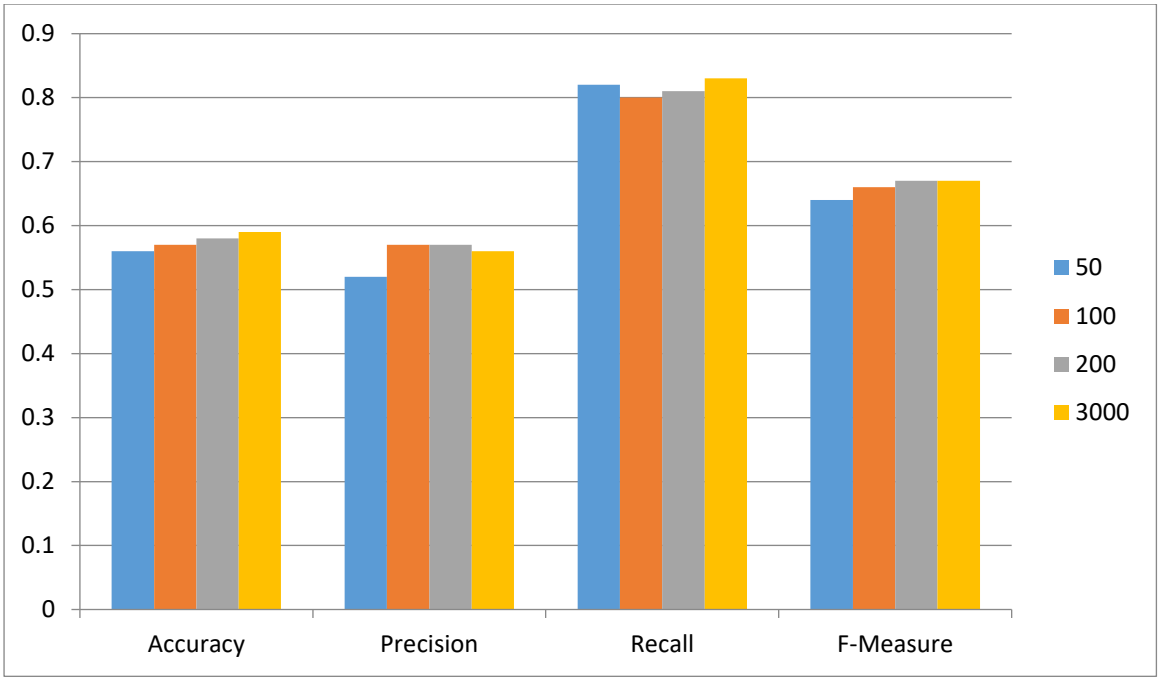

Figure 1: Naive Bayes

Analysing the result we can say that Naïve Bayes approach comparatively performs better when number of features are increased. Considering time complexity and performance Naïve Bayes performs better than other supervised approaches. But precision of this approach is abysmal.

5.1.2.2 Logistic Regression

Accuracy of this approach is 59% for 200 features, 60% for 100 features, 60% for 50 features and 60% for 25 features. Precision of this approach is 93% for 200 features, 95% for 100 features, 96% for 50 features and 98% for 25 features. Recall of this approach is 60% for 200 features, 60% for 100 features, 60% for 50 features and 60% for 25 features. F- Measure of this approach is 0.73 for 200 features, 0.74 for 100 features, 0.75 for 50 features and 0.75 for 25 features.

Table 3: Result of Logistic Regression Approach

Result Name	Number of Feature (200)	Number of Feature (100)	Number of Feature (50)	Number of Feature (25)
Accuracy	59%	60%	60%	60%
Precision	93%	95%	96%	98%
Recall	60%	60%	60%	60%
F-Measure	0.73	0.74	0.75	0.75

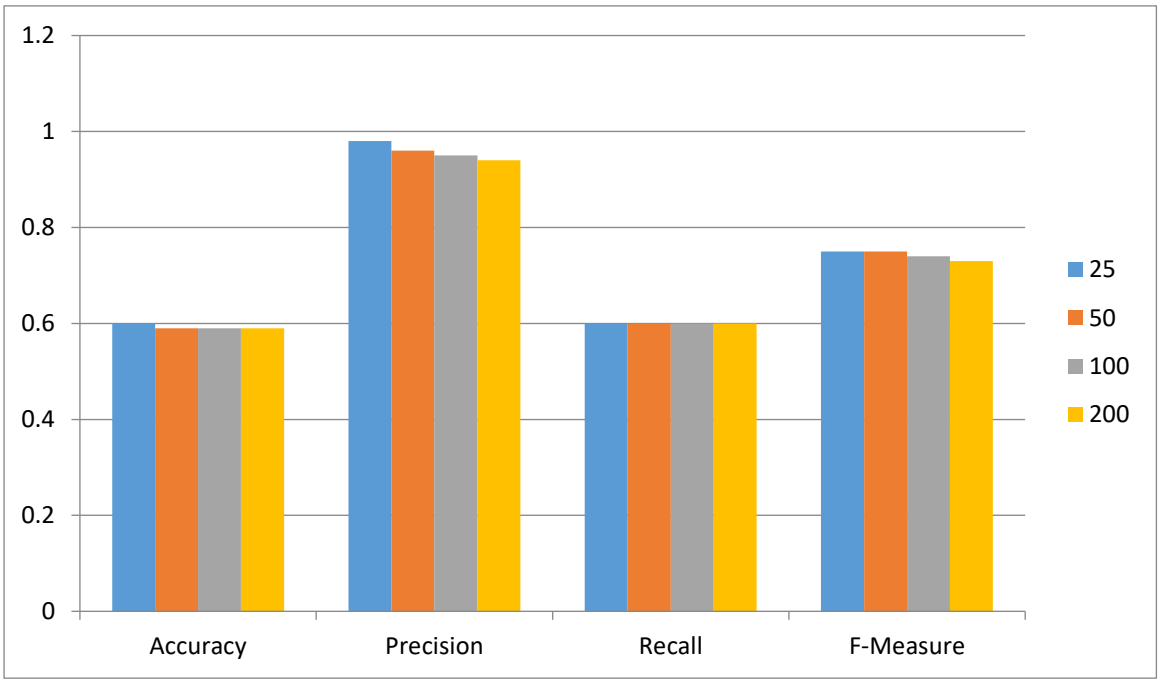

Figure 2: Logistic Regression

Analysing the result we can say that logistic regression comparatively performs better when number of features are decreased. Accuracy and precision of logistic regression is good than other supervised approach but recall is not good enough. The main drawback of this approach is it take so much time when number of features are so much.

5.1.2.3 Decision Tree

Accuracy of this approach is 30% for 100 features, 33% for 50 features and 43% for 10 features. Precision of this approach is 41% for 100 features, 45% for 50 features and 52% for 10 features. Recall of this approach is 38% for 100 features, 46% for 50 features and 68% for 10 features. F- Measure of this approach is 0.4 for 100 features, 0.45 for 50 features and 0.59 for 10 features.

Table 4: Result of Decision Tree Approach

Result Name	Number of Feature (100)	Number of Feature (50)	Number of Feature (10)
Accuracy	30%	33%	43%
Precision	41%	45%	52%
Recall	38%	46%	68%
F-Measure	0.4	0.45	0.59

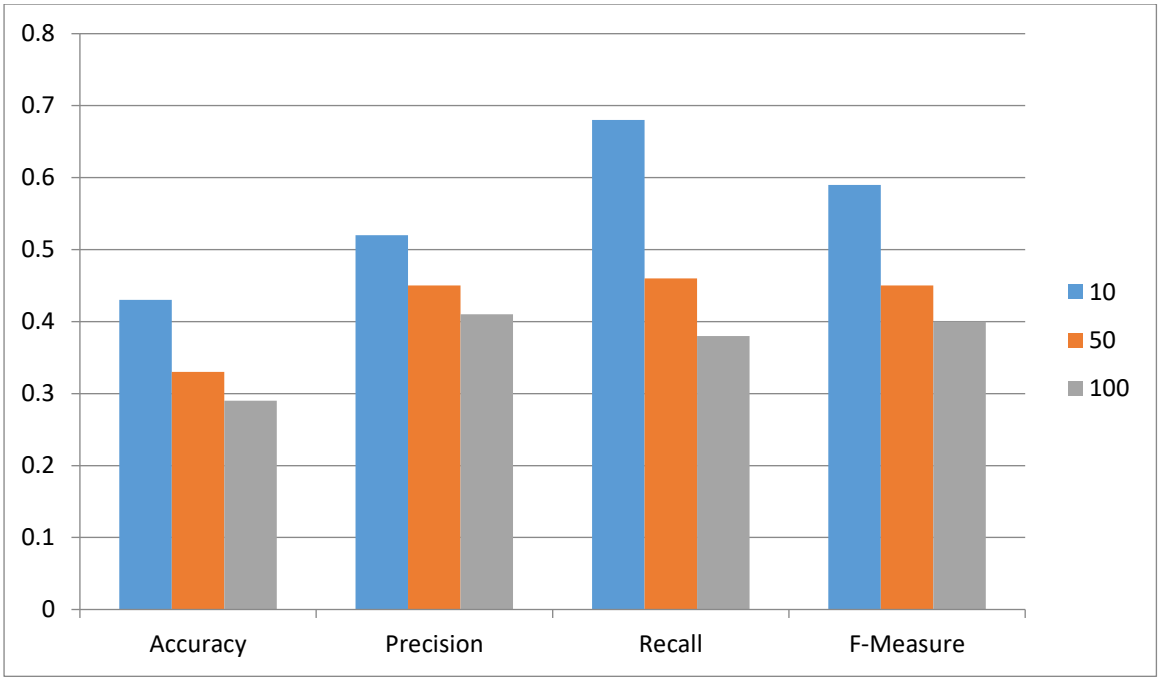

Figure 3: Decision Tree

Analysing the result we can say that decision tree is not suitable for text classification because complexity of decision tree is huge when number of features is huge. Time complexity is also huge in decision tree.

5.1.3 Semi-Supervised Approach

In semi-supervised learning we have used self-learning approach. Here at first we took one fourth data to train the Naïve Bayes classifier initially after labelling them. Then we gradually classified and trained them with a small chunk of unlabelled data. Here we have also checked the performance for different chunk size.

Accuracy of this approach is 54% for chunk size 20, 54% for chunk size 50, 56% for chunk size 80 and 55% for chunk size 100. Precision of this approach is 58% for chunk size 20, 60% for chunk size 50, 61% for chunk size 80 and 61 % for chunk size 100. Recall of this approach is 75% for chunk size 20, 74% for chunk size 50, 75% for chunk size 80 and 74 % for chunk size 100. F-Measure of this approach is 0.65 for chunk size 20, 0.66 for chunk size 50, 0.68 for chunk size 80 and 0.67 for chunk size 100.

Table 5: Result of Self-Learning Approach

Result Name	For chunk size 20	For chunk size 50	For chunk size 80	For chunk size 100
Accuracy	54%	54%	56%	55%
Precision	58%	60%	61%	61%
Recall	75%	74%	75%	74%
F-Measure	0.65	0.66	0.68	0.67

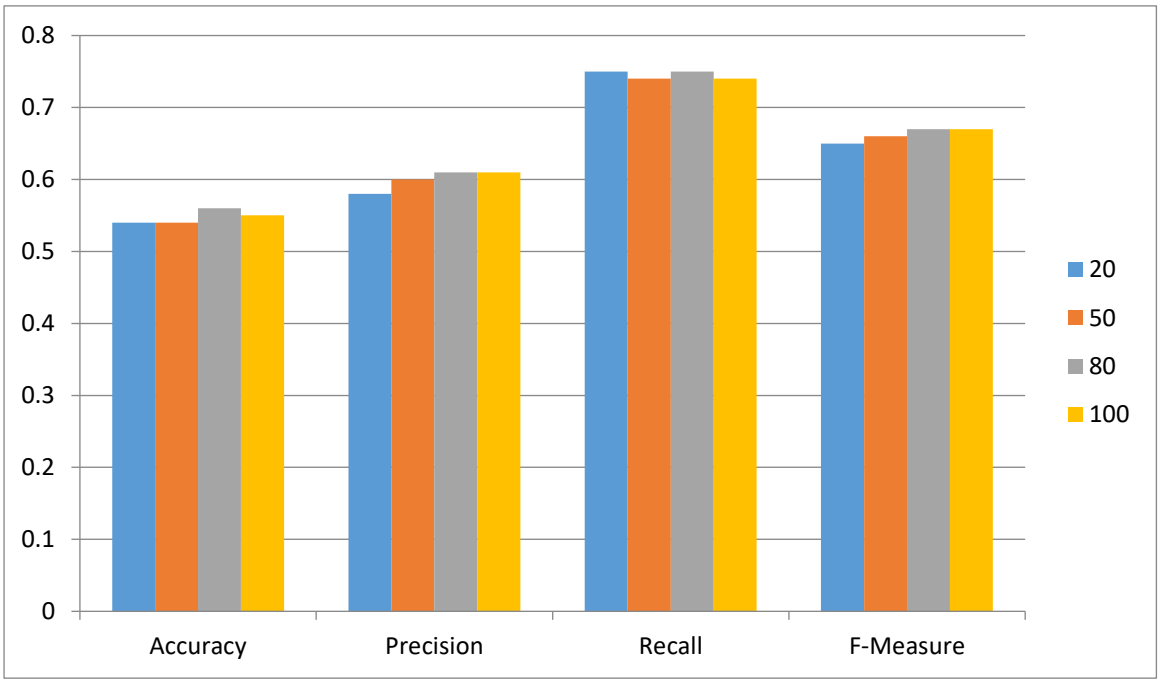

Figure 4: Self-Learning

Analysing the result of this approach we can say that the overall performance of this approach is increased with chunk size but it gives best performance at chunk size 80 after that again its performance degrade.

5.2 Comparison of different approaches

From the following figure 5 we can conclude that similarity histogram based clustering gives more accuracy. The result of F-Measure is also good in similarity histogram based clustering. Precision is so good in logistic regression. In recall similarity histogram based clustering and Naïve Bayes give good result. So, considering all result we can say that similarity histogram based clustering gives comparatively good result.

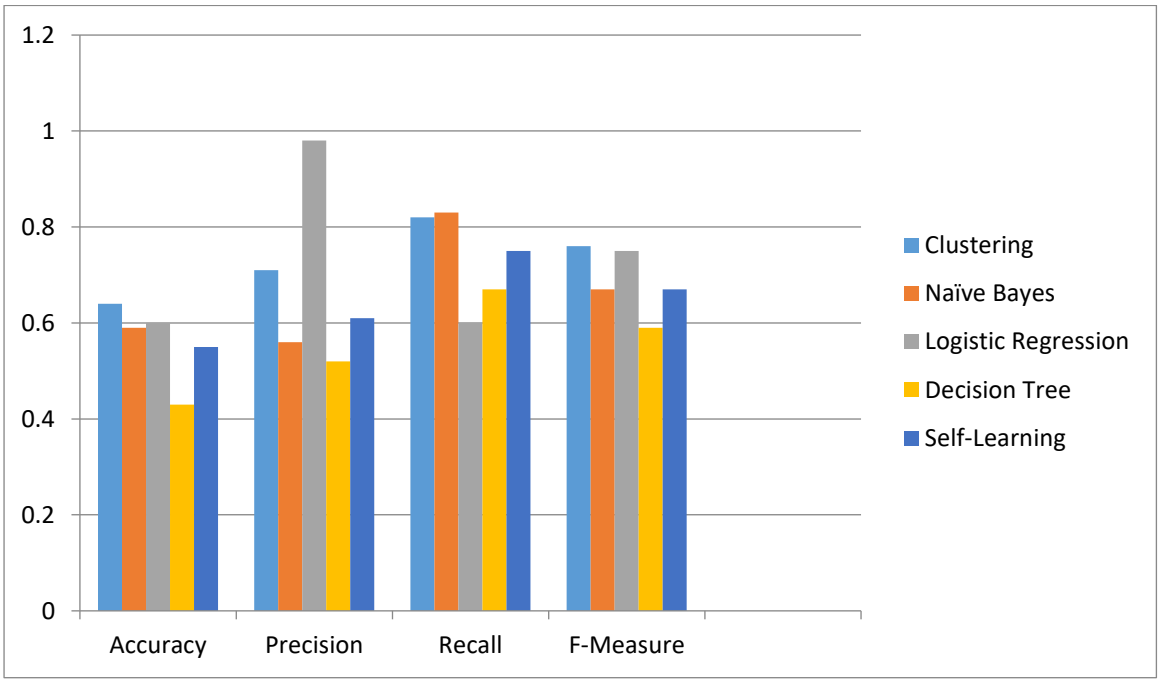

Figure 5: Comparison of Different Approaches

6 Conclusion & Future Scope

The aim of this research was to analyse which machine learning approach perform better for bug classification. For this comparative analysis we have used same dataset. From this comparison we have found that similarity histogram based clustering performs better. The logistic regression also gives good result and logistic regression gives more precision than other approaches. The following approaches will perform better it the dataset contain more precise and accurate information. For supervised learning, labelling is very important but labelling of this dataset was not so good. To obtain good result in supervised learning we need to provide more accurate labelling.

Here, we have make comparison based on only one dataset. In future we could compare based on multiple dataset and we can use advance machine learning approaches such support vector machine, deep learning etc.

7 References

1. Nagwani, Naresh Kumar, and ShrishVerma. "CLUBAS: an algorithm and Java based tool for software bug classification using bug attributes similarities." *Journal of Software Engineering and Applications* 5.06 (2012): 436.
2. Antoniol, Giuliano, et al. "Is it a bug or an enhancement?: a text-based approach to classify change requests." *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*. ACM, 2008.
3. Javed, Muhammad Younus, and HufsaMohsin. "An automated approach for software bug classification." *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*. IEEE, 2012.
4. Dommati, Sunil Joy, Ruchi Agrawal, and S. Sowmya Kamath. "Bug Classification: Feature Extraction and Comparison of Event Model using Na\" ive Bayes Approach." *arXiv preprint arXiv:1304.1677*(2013).
5. Li, Zhenmin, et al. "Have things changed now?: an empirical study of bug characteristics in modern open source software." *Proceedings of the 1st workshop on Architectural and system support for improving software dependability*. ACM, 2006.
6. Sarkar, Kamal. "Sentence clustering-based summarization of multiple text documents." *International Journal of Computing Science and Communication Technologies* 2.1 (2009): 325-335.
7. Xuan, Jifeng, et al. "Automatic bug triage using semi-supervised text classification." *arXiv preprint arXiv:1704.04769* (2017).
8. Ohira, Masao, et al. "A dataset of high impact bugs: Manually-classified issue reports." *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*. IEEE, 2015.
9. http://ambari.apache.org/
10. http://camel.apache.org/
11. http://db.apache.org/derby/
12. http://wicket.apache.org/