IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 6, Ver. I (Nov.- Dec. 2017), PP 20-24
www.iosrjournals.org

Text Classification: Improved Naive Bayes Approach

Amritesh Srivastava, Rashik Bhasin
Indian Institute of Technology, Delhi; Jaypee Institute of Information Technology, Noida

Abstract: *Automatic text classification is becoming increasingly vital nowadays. The two most common approaches for text classification using Naive Bayes Approach consider either Multivariate Bernoulli model or Multinomial model. The Multinomial model is usually observed to give better results for large number of features. We suggest some improvements in the Multivariate Bernoulli Naive Bayes Approach. We were able to achieve an average accuracy of about 99% as opposed to about 85% using basic Multivariate Naive Bayes Approach. We also reduced the number of features required making the classifier space and time efficient.*
Keywords*: Text classification, naïve bayes, multivariate, bernaulli.*

Date of Submission: 01-11-2017 Date of acceptance: 07-11-2017

I. Introduction

With the increasing use of internet, it has become almost impossible to keep track of all data manually. Since most of the data available on the internet is in the form of text, text classification is becoming increasingly important day by day. The main purpose of text classification is to classify a given text document to a particular category e.g religion, politics, sports etc. The simplest and most popular classifier is based on Naïve Bayes Approach. The Bayes classifier tries to find the class for each document which results in minimum error using probability approach. The naïve Bayes approach assumes that there is no dependence between various features considered for classification. This assumption works quite well for text classification problems. We use the Multi-variate Bernaulli event model for our system. In this model, only presence or absence of a word is considered. The frequency of occurrence is not captured. On the other hand the Multi-nomial model which captures frequency of words also, has been proven to be more accurate for large vocabulary sizes. However we have suggested some modifications in the multi-variate model which results in great performance.

Problems with traditional approach:
Most text classification systems based on Naïve Bayes Approach have been shown to result in an accuracy of about 80-85%. The main limitations of these implementations are
a) Ignoring the frequency of the words.
b) Ignoring the semantics of the data.
c) Inherent error probability involved in Bayes approach.
d) Equal importance to all words.
e) Incorporation of Negative evidence i.e. evidence from words that do not occur in the document[4].

How our approach tries to tackle them:
First, we exclude the negative evidence from the model i.e. we do not consider the absence of a word having any effect on classification of a document. We actually get better accuracy than the traditional model since in text classification presence of a word is a much more important characteristic than its absence. Second, we remove all "stop words" from the feature vector. Stop words are those which have no relation to the category of data. They are usually filtered out for most Natural Language Processing applications. Third, we do not treat each word similarly. There are some words that play a key role in text classification. Throughout this paper, we call them"**golden words**". These are those words which have a very high frequency of occurrence in one class but very low in all other classes. For example the word "speed" occurs much more times in class "autos" than any other class. We give very high weightage to these words for our decision of class.

I. Literature Review

A lot of research has been done in recent years concerning comparison of various text classification techniques and suggesting modifications in Naïve Bayes classifier. Lingling Yuan(2010)[2] implemented an improved version of Multi-variate Naïve Bayes model on the Starter Edition text classification data made by Sogou laboratory which had 9 categories and 17910 documents and succeeded in improving the precision from 80% to 85%. Andrew McCallum et al.[4] compared the performance of Multi-nomial and Multi-variate classifiers. They observed that multi-nomial model is uniformly better than the multi-variate model reducing the average error by about 27%. They also performed the experiment on 20Newsgroups data that is used by us. Moreover they held out 20% of the data for testing, same as us. They claim to achieve an accuracy of 85% using multi-nomial approach and 74% using multi-variate approach if we consider the maximum vocabulary sizes. They also concluded that multi-variate model handles large vocabularies poorly as compared to multinomial model. Hetal Doshi et al.(2011)[3] also studied the results of multi-nomial model on 20Newsgroups data. They divided the datasets into two groups-one consisting of similar categories and other unrelated categories. They achieved a maximum accuracy of 86.32% on 1st group and 97.48% on the second. However they conducted their experiment only for two categories of documents. The results are expected to vary if we increase the number of categories to 7 or 8.

II. Experimental Results

We first implemented the basic Naïve Nayes model based on Multi-variate approach and then tried various modifications on it.

Data sets and protocol: We used the 20newsgroup dataset available through UCI data repository. For this experiment, we have picked up a total of 7230 articles belonging to 8 newsgroups corresponding to rec.* and talk.*. We implemented multi-variate Naïve Bayes algorithm in MATLAB to classify each article into one of the newsgroup categories. We generated the training and test data to perform 5-fold cross validation by randomly dividing our data into 5 equal sized splits. We trained on each possible combination of 4 splits and tested on the remaining one. From the resulting Confusion Matrix, we calculated average accuracy for all classes by dividing sum of all diagonal elements by sum of all elements in it. We repeated the experiment for all possible combination of 4 splits and calculated overall average accuracy by taking an average of all 5 accuracies observed. We tried three approaches and studied the effect of various modifications on overall accuracy of the classifier:

1.Basic Approach: We implemented basic multivariate Naïve Bayes model. We use as a feature vector each unique word occurring in all the documents giving a total of 37061 features. However, we later ignored those words whose total frequency of occurrence was less than 8 which accounts to about 0.0015% of total words in the dataset. This did not have any negative impact on accuracy of classification. Moreover we were left with only 7829 features which means better efficiency and much less space required. These words have no impact on our classification decision, since they occur very rarely and hence probability of finding them again in a test set is also very low. Even if they do appear in a test set, they carry a very little information of the class they belong to since they have very low probability of occurrence in any class. Thus although this seems to be a very simple modification, it reduces our time and space requirements by a factor of about 5.Finding the best vocabulary size giving optimum accuracy in minimum time is an interesting topic for future work. We made some modification in the standard probability function corresponding to Bernaulli distribution also. The posterior probability of category Ci given the value of a feature vector Xj is:

$P(Ci/Xj)=Xj*P(Xj/Ci)+(1-Xj)*(1-P(Xj/Ci))$ (1)

The probability of category Ci given feature vector X for Naïve Bayes model is

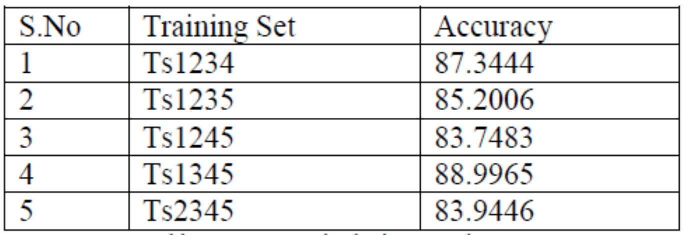
P(Ci/X)= P(Ci)□*I P*(*Cj* / *Xi*) (2)

Since we noticed that presence of a word is much more important in text classification than absence of a word, we removed the second term in the equation (1). We also implemented the Laplace correction i.e. we added 1 to frequency of each word so as to avoid 0 probabilities which could have collapsed equation (2).

Since the only use of equation (2) is for comparison for various categories, we were free to perform many modifications which would otherwise have been impossible. We take log of both sides of equation 2. Now we have to maximize

$P(Cj/X)=\log(P(Cj))+\Sigma\log(2*Xi*P(Xi/Cj))$

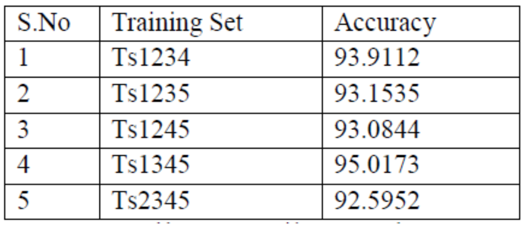
where Xi can only be either 0 or 1 depending on whether the word Xi appears in the document or not. Since the value of P(Xi/Cj) lies between 0.5 and 1. Value of log lies between 0 and 0.3. Here we did not calculated log if Xi was 0 which ignored absurd values of log and also resulted in tremendous performance gain. Since now we calculate log only for words occurring in the test data instead of for every word in the feature vector. The resulting accuracies for all five training datasets with the above modifications is summarised in the table below:



S.No	Training Set	Accuracy
1	Ts1234	87.3444
2	Ts1235	85.2006
3	Ts1245	83.7483
4	Ts1345	88.9965
5	Ts2345	83.9446

Table 1. Accuracy using basic approach
Average Accuracy=85.84%

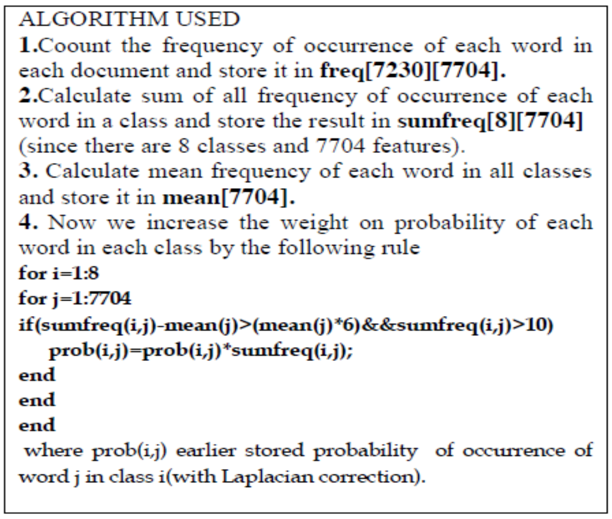
2.Remove Stop words: Removing the stop words has been an integral part of many Machine Learning algorithms for years. We took a comprehensive list of 635 stop words from various sources on the internet and removed them from our feature vector. We got 186 hits in our feature vector. After removing them we applied the same model as above and observed a dramatic increase in accuracy to about 93%. Here we also propose an automated method to remove such words. We just need to calculate the sum of frequency of occurrence of each word in each class. The words for which sum of frequency in each class is close to each other i.e. there is a small variance, can be considered stop words and hence can be removed from our feature vector. Considering these features might prove useful for some other classifiers possibly using continuous data. But in text classification, where occurrence of each word is being considered for classification decision, the inclusion of these words does not provide any useful insight to document's class. Rather it may lead to a decrease in overall accuracy. After removing these words and again performing the experiment with our training data, we got the following results:



S.No	Training Set	Accuracy
1	Ts1234	93.9112
2	Ts1235	93.1535
3	Ts1245	93.0844
4	Ts1345	95.0173
5	Ts2345	92.5952

Table 2. Accuracy with no stop words
Average Accuracy=93.55%

3.Golden words: We further noticed that there are some words which play a very crucial role in deciding the class for a document. We call them "golden words" since they can lead to a great increase in accuracy and if properly identified, they can pay a lot of contribution to the classification decision.
These words are quite easy to recognise. Their probability of occurrence in one class is much higher than all other classes e.g. consider the word "speed" which occurred 117 times in the class "autos" and 133 times in total. Please note that we considered total frequency of occurrence of words for this part instead of just boolean values since they seemed to provide more information. We increased the weight on probability of each word in each class using the following algorithm. Please note that prob (i,j) is not necessarily a probability measure anymore but it doesn't make a difference since its only purpose is to act as a weight to our classification. And since we multiplied it by "sumfreq", now the more skewed the presence of a word is in one class and the more number of times it occurs in that class, the more is its participation in decision making. For example in some cases we changed prob(i,j) from 0.5076 to 25.41. Since we are taking log, it does not affect our decision drastically.



ALGORITHM USED

1.Coount the frequency of occurrence of each word in each document and store it in **freq[7230][7704].**

2.Calculate sum of all frequency of occurrence of each word in a class and store the result in **sumfreq[8][7704]** (since there are 8 classes and 7704 features).

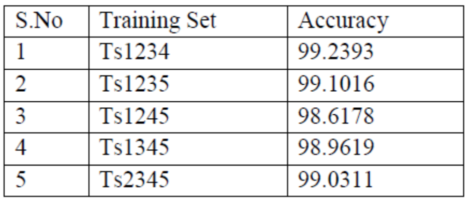
3. Calculate mean frequency of each word in all classes and store it in **mean[7704].**

4. Now we increase the weight on probability of each word in each class by the following rule

```
for i=1:8
for j=1:7704
if(sumfreq(i,j)-mean(j)>(mean(j)*6)&&sumfreq(i,j)>10)
    prob(i,j)=prob(i,j)*sumfreq(i,j);
end
end
end
```

where prob(i,j) earlier stored probability of occurrence of word j in class i(with Laplacian correction).

Thus the overall purpose of the above step is to consider each word on its merit rather than giving each word same importance. Some words may contribute about 10 times some other words. We found 881 such words, and after the above modification, we noticed an amazing increase in accuracy. The results are given below:



S.No	Training Set	Accuracy
1	Ts1234	99.2393
2	Ts1235	99.1016
3	Ts1245	98.6178
4	Ts1345	98.9619
5	Ts2345	99.0311

Table 3. Accuracy using golden words

Average Accuracy=98.99%

III. Results And Conclusions

The accuracies of all the approaches tried is depicted in the graph below:

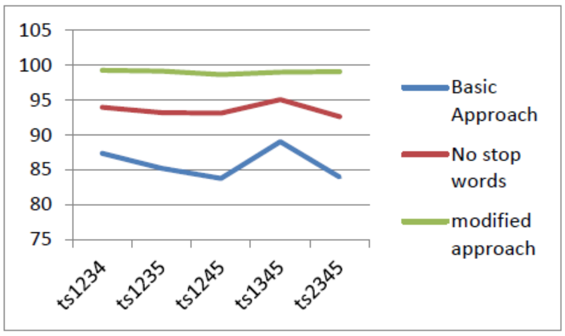


Figure 1: Accuracies of various approaches

We noticed a dramatic increase in accuracy of classification using our approach. This can be attributed to allocating different weights to different words. Finding the words that can be more significant than the others in classification can have numerous applications. We just have to sort the words according to their relative importance in classification decision. We can limit the size of feature vector theoretically to any small number that can be easily fitted into any limited memory making it ideal for internet applications. This can also pave way for real time classification since we can finish our classification task in any given amount of time. The variation of accuracy and time taken depending on number of features taken can be a topic of further research.

IV. Future Prospects

There are numerous ideas and approaches that can be implemented to obtain better results and most importantly to test the generalization of the system-

1. Perform the same experiment with different benchmark datasets.
2. Study the variation of accuracy and time taken with number of features.
3. Improve the feature selection process so that it can be general enough to be used in various applications.
4. Study the variation of accuracy with size of training set.
5. Comparative study of various text classification techniques with our approach.

References

[1] Andrew McCallum And Kamal Nigam, A Compariison of Event Models for Naïve Bayes Text Classification, Pittsburgh,PA 15213.

[2] Lingling Yuan, An Improved Naive Bayes Text Classification Algorithm In Chinese Information Processing,Henan Polytechnic University, Jiaozuo, China, 2010.

[3] Hetal Doshi and maruti Zalte, Performance of Naïve Bayes Classifier – Multinomial Model on Different Categories of Documents, Vidyavihar, Mumbai.

[4] Karl-Michael Schneider,On Word Frequency Information and Negative Evidence in Naive Bayes Text Classifications, University of Passau, Germany.

Amritesh Srivastava Text Classification: Improved Naive Bayes Approach." IOSR Journal of Computer Engineering (IOSR-JCE) , vol. 19, no. 6, 2017, pp. 20-24