Text Classification using Naive Bayes

For this problem, the dataset is the Large Movie Review Dataset. Given a movie review, the task was to predict the sentiment of the review as either positive or negative.

Number of Training samples are 25000 reviews

Number of Testing samples are 15000 reviews

a. Implementing a Naive Bayes Classifier

In the first part, we have implemented the model on original data. The reviews are first pre-processed only to remove separators of the words and punctuations.

The dictionary is created by adding all the words that appear at least once in the data.

This is also called ”Bag of Words” model.

We also keep a count of the frequency of the words in the data in the dictionary. Laplace Smoothing is done by using C = 1 to control the strength of the smoothing i.e., to avoid any zero probabilities for the words that didn’t occur in the train set.

$$\theta_{l/k} = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=k\} \sum_{j=1}^{n^{(i)}} 1\{x_j{}^{(i)}=l\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)}=k\} \cdot n^{(i)} + |V|}$$

$$\phi_k = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=k\}}{m}$$

Above are the maximum likelihood of the parameters used for training the model.

We have used log in our implementations to avoid underflow issues.

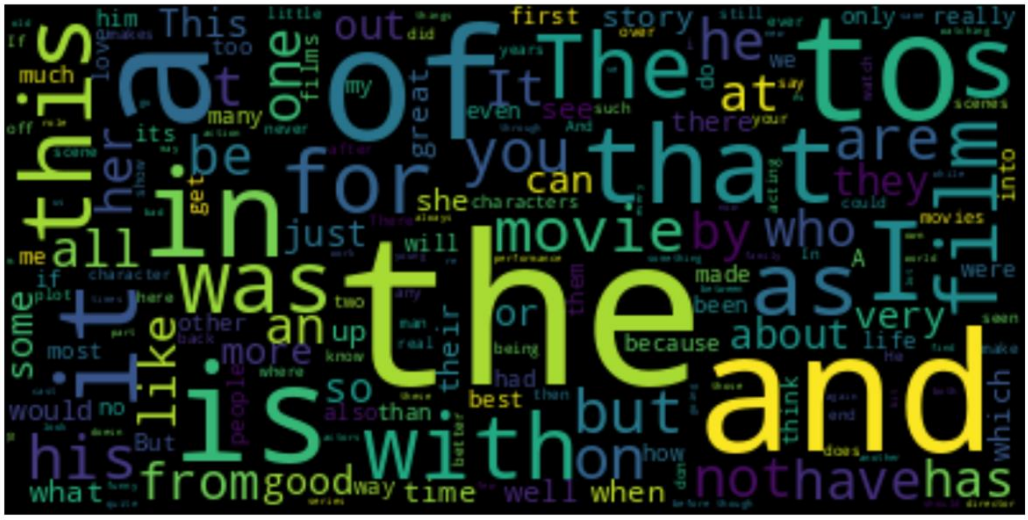
Results

i. Accuracy got over the train set = 90.800%

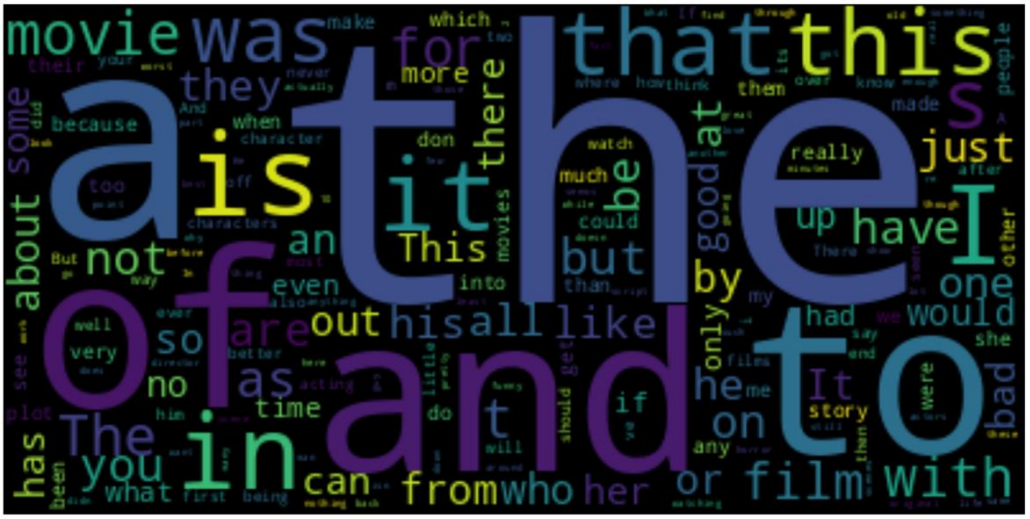
Accuracy got over the test set = 82.307%

ii. Below are the word clouds obtained for each positive and negative class.

Positive class word cloud



Negative Class word cloud



Word clouds are repressing which words have maximum frequency in that particular class. As in this part we haven't use stopword removal we are getting words like 'a', 'the', 'and', 'of' with maximum frequency.

b. random and positive baseline predictions

We are getting accuracy for both predictions using test data as follows -

i. The test accuracy for random predictions=49.480%

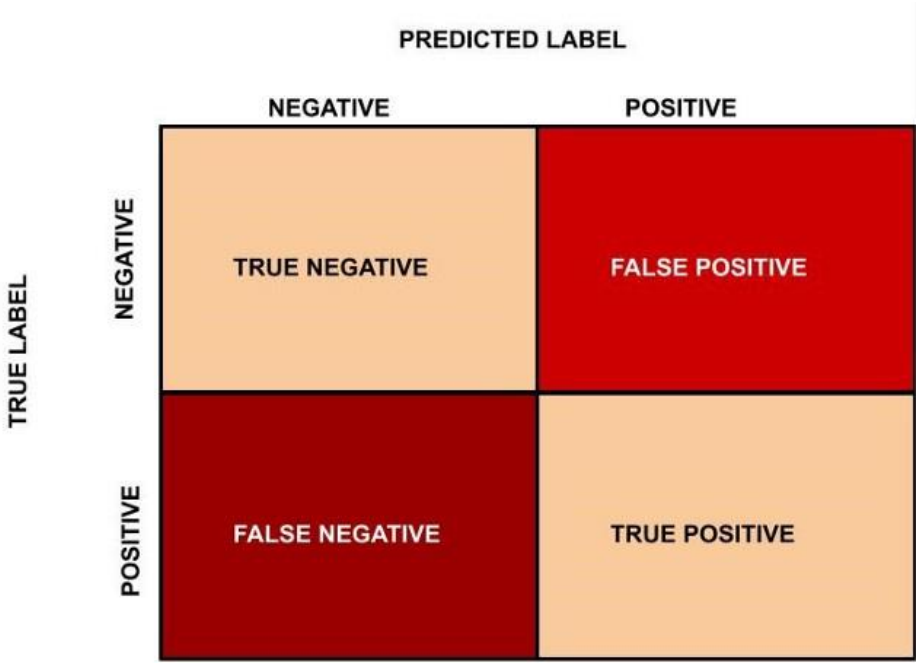
ii. The test accuracy for predicting all values positive=66.667%

iii. Improvement of our algorithm over the random baseline = 82.307/49.480 = 1.66 times better

Improvement of our algorithm over the positive baseline = 82.307/66.667 = 1.23 times better

c. Confusion matrix

For our problem, as there are two classes we will use binary confusion matrix. The following diagram illustrates the confusion matrix of binary classification problem.



As per this diagram following are the confusion matrix for part a & b.

- Confusion matrix for original test data

  [[4160 840]

  [1814   8186]]

  For this matrix, The highest value of the diagonal entry is the positive class with value 8186.

  Which means that our model is able to predict positive class more accurately.

  The same can also say because false positive value is less than false negative.

- Confusion matrix for random predict data

  [[2470 2530]

  [5066 4934]]

  In this matrix, The highest value of the diagonal entry is the negative class with value 5066.

  which indicates, using random prediction negative classes are predicted slightly accurate than positive classes.

- Confusion matrix for positive predict data

  [[0 5000]

  [0 10000]]

  The highest value of diagonal entry is positive class with 10000.

  From this matrix it is clearly visible that it only predicting the positive class.

d. Stemming and Stopword Removal

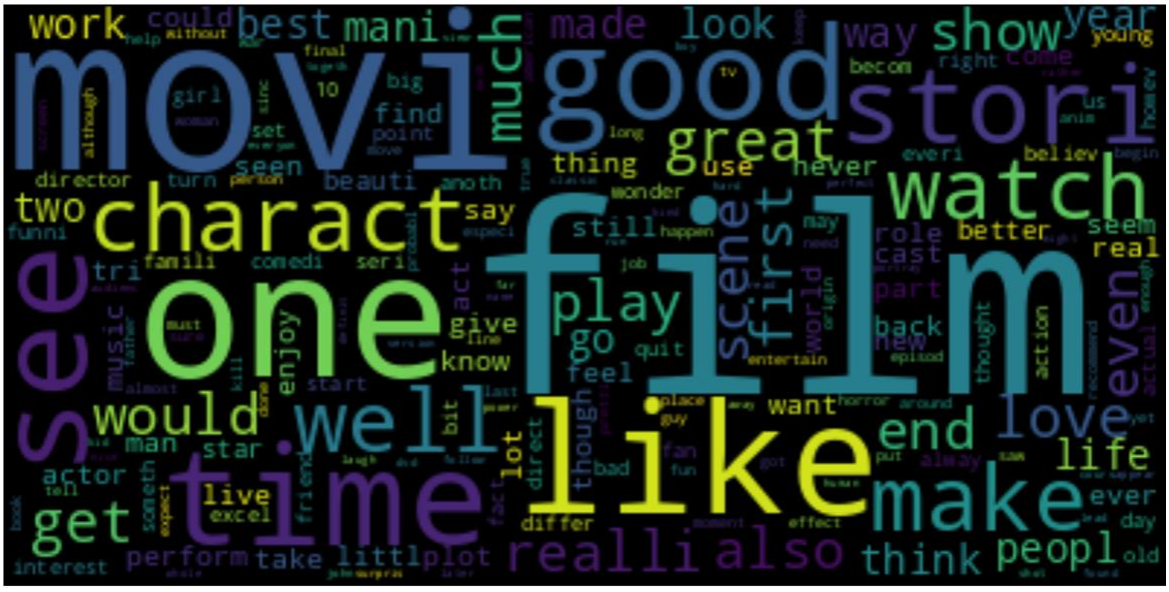
i. In this step we are filtering the data by firstly, removing stopwords and after that applying stemming for each words.

Here, for removing stopwords we have used *stopwords* from *nltk.corpus* library. And for stemming each words we have used *PorterStemmer* from *nltk.stem*.

After transforming the data we will again find wordcloud for each class.

ii. Followings are the wordclouds.

Positive Worldcloud with transformed data



Negative Worldcloud with transformed data

iii. In this part learning a new model using this transformed data.

And obtained the test set accuracy = 83.400%

iv. The accuracy of test set after transforming the data is slightly better than the model without stemming.

This is because, before there were so many stopwords with max frequencies. Because of those accuracy was slightly less.

e. Feature engineering

Feature Engineering is an important aspect in Machine Learning which is used to manipulate existing features or to create new features to improve the overall accuracy of the model.

In the previous models, we have used every word in the review to create the dictionary hence every word is used as a feature.

i. Using word based bi-grams, to combine consecutive two words as a single feature.

For doing this we have used *ngrams* from *nltk.util*

used ngrams with n=2 for bi-grams

after learning the model using bigram data, the test accuracy we have

got = 83.840%

which is almost same with accuracy after stemming.

ii. now using ngram with n=3 for tri-grams

we are using this to add one extra feature.

After that trained the model with this trigram data.

Accuracy we got = 75.120%

which is less than original test accuracy. Because with trigrams, it may that no of feature vector decreasing but number of unique feature vector increase very much, because of this accuracy became less.

iii. As per above points, while using bigrams model gives more accurate prediction than using trigrams.

f. Computing Precision, Recall and F1-score

For computiong precision, recall and F1 score we need the confusion matrix.

i.

if confusion matrix is ,

[[TN FP]

[FN TP]

then Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

F1 Score = 2* (Precision * Recall)/(Precision + Recall)

part a .

confusion matrix,

[[4160 840]

[1814 8186]]

Precision = 8186/(8186+840) = 0.907

Recall= 8186/(8186+1814) = 0.819

F1 Score = 2* (0.907* 0.819)/(0.907 + 0.819) = 0.8607

part d.

Confusion matrix,

[[4105 895]

 [1595 8405]]

Precision = 8405/(8405+895) = 0.904

Recall= 8405/(8405+1595) = 0.840

F1 Score = 2* (0.904* 0.840)/(0.904 + 0.840) = 0.871

part e.

Confusion matrix,

- bi-gram data

  [[4508 492]

   [1932 8068]]

  Precision = 8068/(8068+492) = 0.942

  Recall= 8068/(8068+1932) = 0.807

  F1 Score = 2* (0.942* 0.807)/(0.942+ 0.807) = 0.869

- tri-gram data

  [[4023 977]

   [2755 7245]]

  Precision = 7245/(7245+977) = 0.881

  Recall= 7245/(7245+2755) = 0.724

  F1 Score = 2* (0.881* 0.724)/(0.881+0.724) = 0.79

ii.

According to above values we got, we can say the model after stemming and stopword removal is more suited for this kind of dataset.

As in this dataset common words between reviews are very less, due to that when we use additional set feature vector number of unquie feature vector is increasing and frequency of those also decreasing.

Due to this prblem, accuracy decreasing while using additional feature vector.