BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

FIRST SEMESTER - 2017-18
MACHINE LEARNING
ASSIGNMENT 1 FINAL REPORT

NIKHIL GOEL
GROUP-43
2015A8PS0385P

<u>MOVIE REVIEW CLASSIFICATION(SENTIMENT ANALYSIS)</u>

Problem Statement

Given a labelled data set of movie reviews, classification of the movie review as good or bad. (Binary classification)
Movie reviews in the training/testing set are given in words with rating(Label) - 1 to 10

Data description

Large Movie Review Dataset v1.0 (Imdb movie review data set).
The dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg).

There are no more than 30 reviews for any movie because reviews of the same movie are highly correlated. Further, the train and test sets contain a disjoint set of movies, so no significant performance is obtained by memorizing movie-unique terms and their associated with observed labels. In the train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Thus reviews with more neutral ratings are not included in the train/test sets.

The average length of a negative movie review in training set is 230.9 while the average length of a positive movie review in training set is 234.4.

There are a total of 89527 unique words in all of the training set.

Each movie review is in a separate text file which is named with the convention [[id]_[rating].txt] where [id] is a unique id and [rating] is the star rating for that review on a 1-10 scale. For example, the file [train/pos/200_8.txt] is the text for a positive-labeled train set example with unique id 200 and star rating 8/10 from IMDb.

Procedure

The first step is preprocessing the movie reivew. This may include -

- Lower casing all the words present in the review
- Removal of numbers and replacing it with a token.
- Word stemming where similar words are replaced by a single word.

Creating the vocabulary list where we choose the words which will be used for classification and the words that will be ignored. The proper nouns which includes names of characters of movies or the movie title itself should be excluded from the vocabulary list. This is done as to reduce the number of attributes. One way is to choose the frequently occurring words. Other way can be using PCA on all the unique words to reduce the number of attributes.

Third step is to create a feature vector of the review. The feature vector can be one hot or dense.

Fourth step is to build a linear classifier. The classification algorithm can be SVM.

Applications

A movie classifier is a basic sentiment analysis framework. It can be used to recommend movies, to understand what makes a movie review good or bad. In the future such a framework can be extended on more fine grained sentiment analysis tasks which can be used to classify all sorts of reviews and can also be used for recommendations.

Modelling details

The libraries used for classification are

- **ReadText -** This library is used for loading and reading the dataset in R.
- **tm -** This library is used for data pre-processing.
- **E1071 -** This library contains the SVM model for classification.
- **Dplyr -** It is used for data manipulation.
- **Caret -** It is used for model evaluation.

The first step would be to load the data set in R. Since the data set is text files, readtext iterates over the entire directory and loads the entire text file into data frames with two columns. First column represents the name of the file and the second column has the data inside the text file.
Hence two data frames are made separately for test and train data sets containing 25,000 entries each.

Next, we need to extract the labels from the name of the file as mentioned in data description. This is done using strsplit function. After extracting the labels, we need to create the feature vector which contains labels for all the movie reviews and then concatenate it with the original data frame.

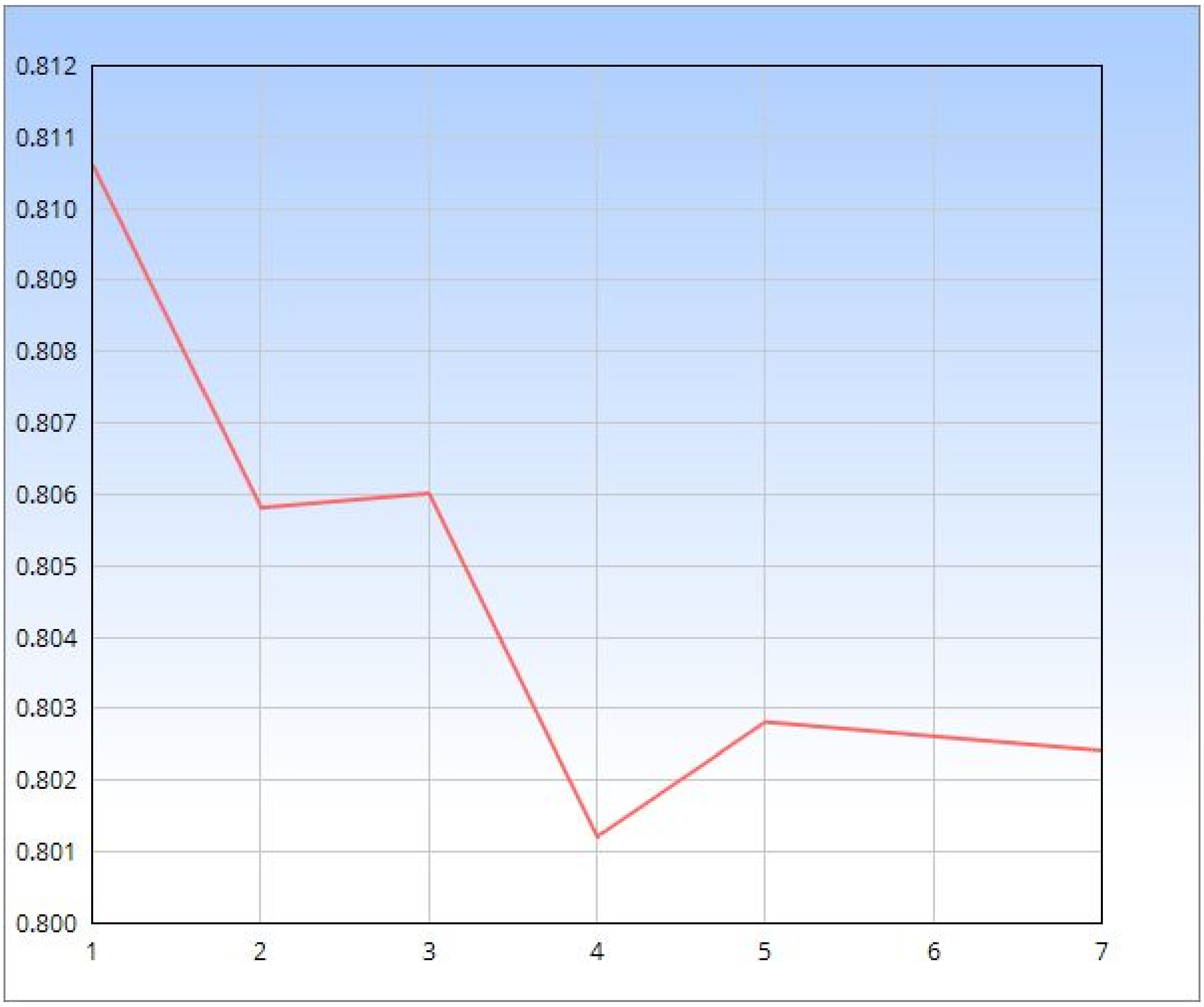
After this, randomization and sampling of data set is done. Bag of words tokenization is created by using the corpus function and creating the corpus of dataset.

Next step would be to preprocess the data. In preprocessing, the text is transformed to lowercase, punctuations are removed, numbers are removed, stopwords are removed, and extra white space is deleted. Stemming of words is not done as it was decreasing the accuracy of the model. (Details in results)
After this, DocumentTermMatrix is created which is shows the frequency of words used and also it is a sparse matrix. Frequency word dictionary is created using this documentTermMatrix.

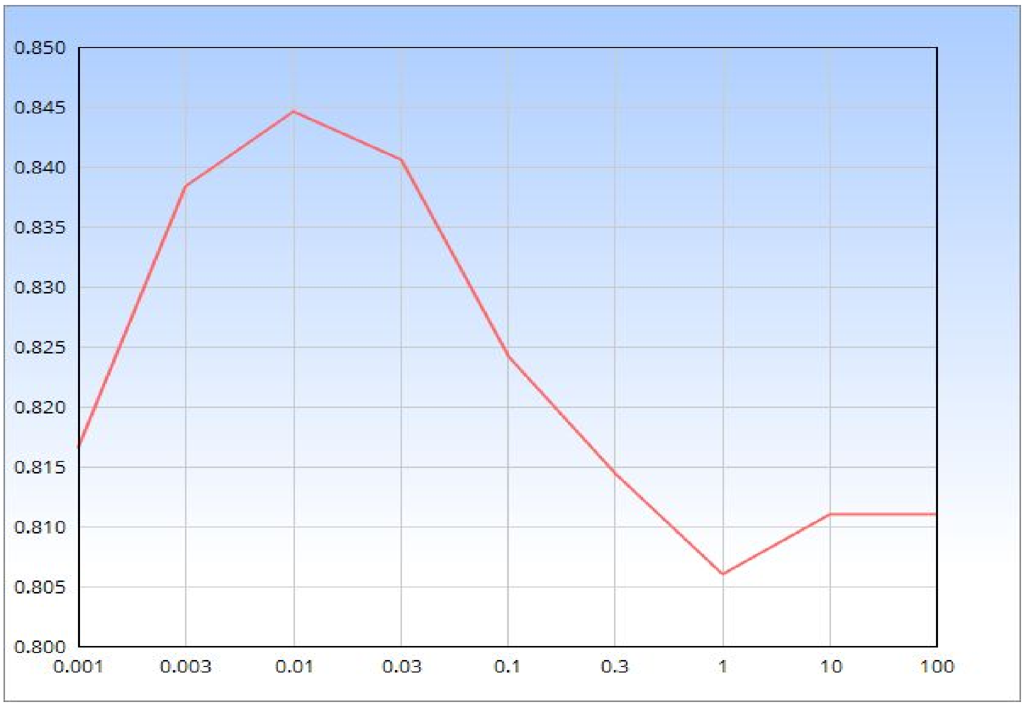
FInally, the model of SVM is trained and results are displayed in confusion matrix. The kernel chosen in linear and the cost is optimized.

Results

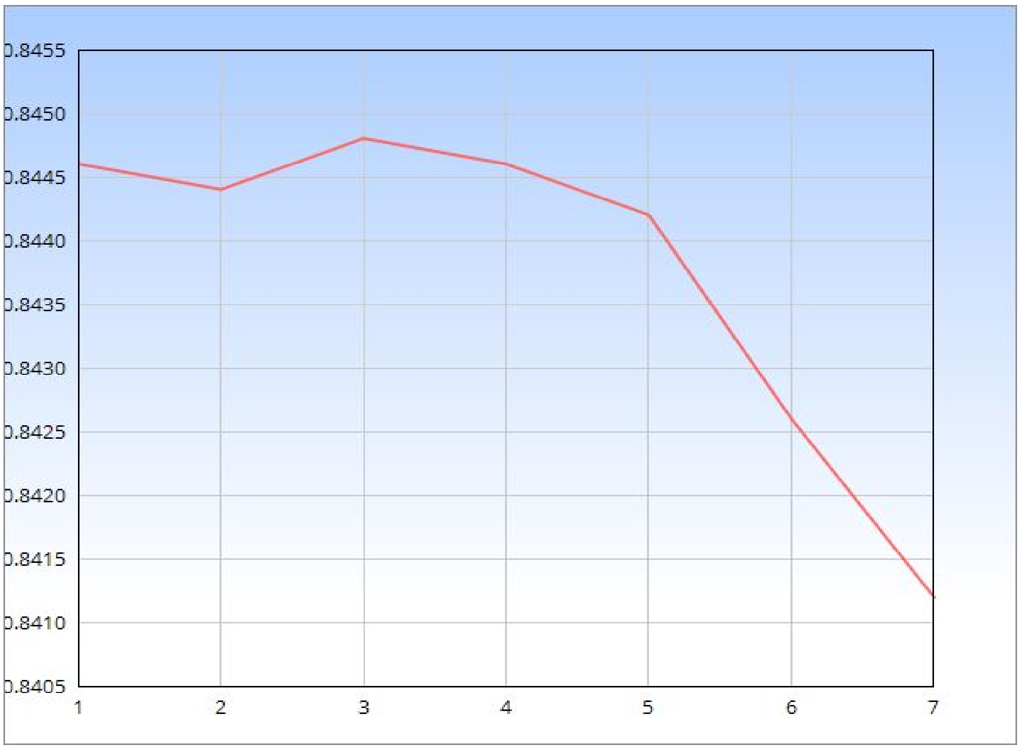
The final accuracy achieved(on optimum parameters) on the test data (25,000 examples) is **<u>0.86.</u>**
Approximately, it took 15 minutes for the model to train. **Hence, all further results are shown on a sample of this data set containing 5000 training and 5000 testing examples.**
The initial accuracy on the sample test data using 5 frequency words and cost in svm as 1 was **<u>0.8028.</u>**
Different number of frequency words were tried and the best frequency achieved with cost is 1 was with frequency word = 1. The plot for the same is shown below.



Cost was started from 0.001 and incremented. The best cost was achieved at **<u>cost = 0.01</u>**.
The plot for the same is shown below.



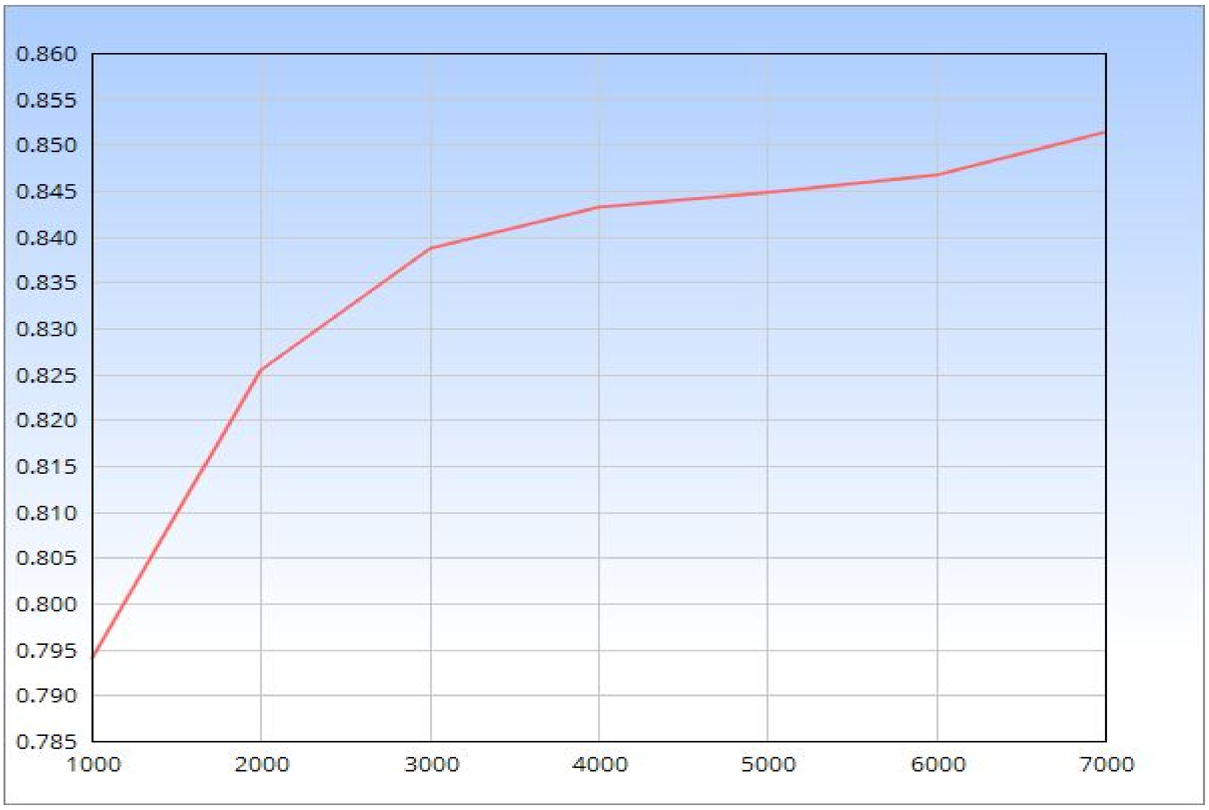
For cost = 0.01, the best accuracy was with **frequency word = 3.** Hence the best accuracy achieved on test data by training the sample dataset with SVM with these parameters was **0.8448**.



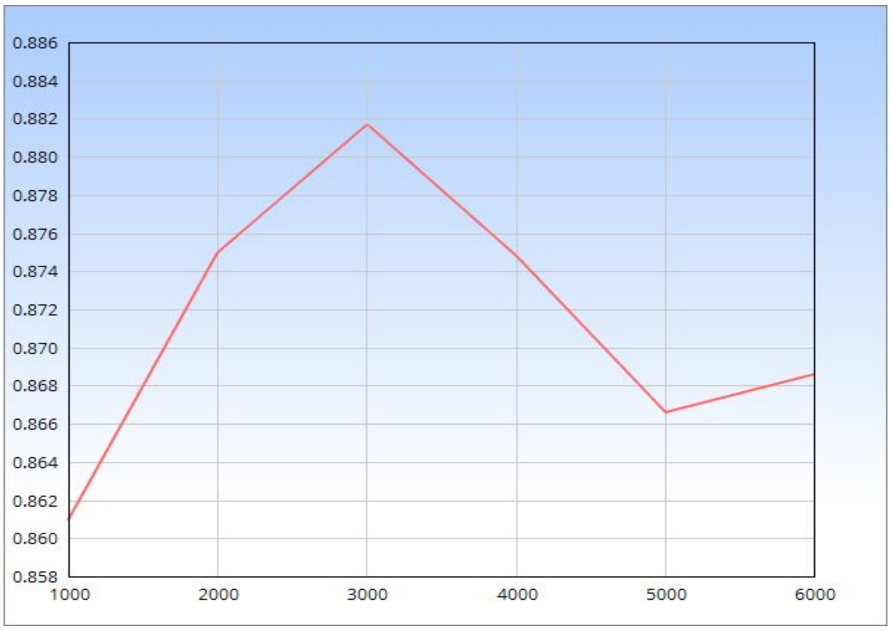
The kernel used in SVM was linear kernel. There was a big drop of accuracy while using Gaussian kernel and Polynomial kernel of degree 3.

Also, using word stemming(PorterStemmer) decreased the accuracy to **0.785** and hence it was not used.

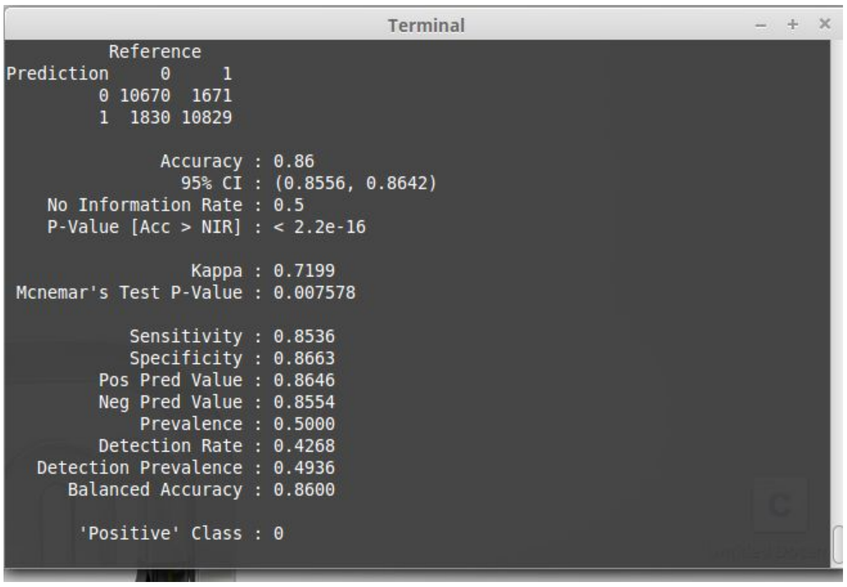
The accuracy is plotted against the number of training/testing(Same number) examples upto 7,000 training and 7,000 test examples. Hence accuracy is linearly increasing as the data set increases but the increment decreases. Hence it shows that after a limit, increasing the dataset would not lead to significant change in accuracy.



Another plot is made which trained the model on the complete training data(25,000) and the testing data changes. The accuracy increase till 3000, and then decreases with minor fluctuations till it reaches 0.86 on the complete test data set of 25,000.



The confusion matrix and the details of the prediction on the entire test data is shown below



```
          Reference
Prediction     0     1
         0 10670  1671
         1  1830 10829

               Accuracy : 0.86
                 95% CI : (0.8556, 0.8642)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7199
 Mcnemar's Test P-Value : 0.007578

            Sensitivity : 0.8536
            Specificity : 0.8663
         Pos Pred Value : 0.8646
         Neg Pred Value : 0.8554
             Prevalence : 0.5000
         Detection Rate : 0.4268
   Detection Prevalence : 0.4936
      Balanced Accuracy : 0.8600

       'Positive' Class : 0
```

Code

Code has been uploaded on github.
https://github.com/nikhilgoel1997/MovieReviewClassification