Sentiment Analysis of Movie Reviews

Shubham Basu[1]*, Chhavi Sharma[2]*, Prahasan Gadugu[3]*

Abstract
Sentiment Analysis is an upcoming field of Machine Learning that comprises Natural Language Processing and Text Analytics. It focuses on extracting useful insights from textual data. In this project, we use textual reviews extracted from Rotten Tomatoes in order to examine and rate the overall polarity of a review from a scale of 0(Most Negative) to 4(Most Positive). We will perform feature extraction using Bag of Words and N-grams models and thereafter implement multiclass classifiers like Naive Bayes, Logistic Regression and Support Vector Machines and perform a comparative study in order to determine the most promising model to gauge the class of a movie review. Our approach of using TF-IDF alongwith Linear Support Vector Classifier gave us the best accuracy of 60 percent.

Keywords
Sentiment Analysis — Feature Extraction — Classifier—Natural Language Processing — NLTK — TF-IDF — Linear SVC — Naive Bayes — N-Grams — KNN — K-Fold — Cross Validation — Pipeline — Kaggle — Rotten Tomatoes — Lemmatization — Logistic Regression

[1] *Computer Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[2] *Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[3] *Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
***Corresponding author**: Manoj Joshi

Contents

Introduction

In the current social media age, online expression has become a virtual measure to rate products and services. This makes Sentiment analysis one of the most upcoming areas in the field of machine learning in order to understand and deal with this proliferation of data in the form of reviews, feedback and recommendations. Sentiment analysis deals with Natural Language processing and text analytics in order to draw two kinds of inferences out of textual data- classification on the basis of subjectivity or polarity. Subjectivity classifies whether a review is subjective or objective, i.e.if the text is manually rated and two different people do the labeling, based on their interpretation, the result of the model and its accuracy will tend to change. On the other hand, polarity refers to classifying the polarity of the data e.g. classifying whether the data is positive, negative or neutral. However, it can become daunting since human language is immensely complex. It can be a herculean task to make a machine analyze and understand grammar, sarcasm, sentence negation, slangs and other such language nuances. Advancements in natural language processing and machine learning make it somewhat possible to analyze such huge amounts of text data and identify user opinions from them.

In this project, we aim to use Sentiment Analysis to determine the underlying sentiment of a movie review on the basis of its textual information. We will try to classify the overall polarity of the reviewers by understanding whether they liked the movie or not from their reviews in order to determine the public opinion of the movie. This project report begins with

the background bringing what has already been done in this field to light. Thereafter we introduce the dataset, and proceed towards the exploratory data analysis and data pre-processing techniques applied in order to gain better insight into the data. We also discuss about the various models and algorithms we investigated and implemented in order to make better predictions. In the end, we present and evaluate our accuracy results and draw comparisons among various models used.

1. Background

Researchers at Stanford University have already worked on movie reviews by using IMDB dataset wherein they used supervised learning to cluster the words with close semantics and created word vectors. The approach was particularly useful when the data had rich sentiment content.

Additionally Bo Pang and Peter Turnkey employed Random Forest Classifiers and SVMs for multiclass classification of moview reviews. However, their papers excluded a neutral category assuming that neutral texts lie closer to classify boundaries and are hard to classify.

2. Data

The dataset used for this project is taken from Kaggle. The data has been collected from Rotten Tomatoes. It includes one pre-labeled training dataset and one unlabeled testing dataset in the form of tab-separated files. The training dataset contains 156,060 data instances and the testing dataset contains 66,292 instances. The train data was parsed from 8500 sentences using Stanford parser. Similarly, the test dataset was parsed from 3310 sentences. Each data instance comprises attributes: “phrase id”, “sentence id”, “phrase”. Each phrase has a phrase Id. Each sentence has a sentence Id. Repeated phrases are included only once in the data. Each phrase in the training dataset is associated with one out of five sentiment labels categorizing movie reviews as follows:

1. 0- most negative
2. 1-somewhat negative
3. 2-neutral
4. 3-somewhat positive
5. 4-most positive

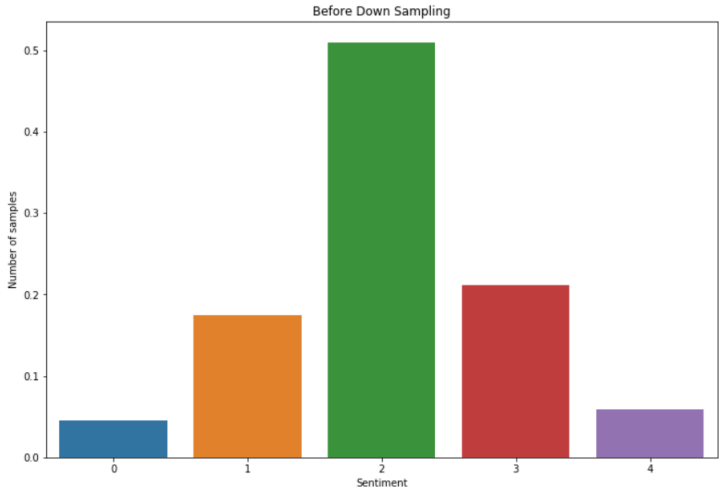
Our data is imbalanced since around 50 percent of the training samples attribute to 2 i.e., neutral. Therefore, we will have to balance our data before classification. Finally, we will carry out cross validation in which the complete dataset will be divided into multiple folds with different samples for training and validation each time and the final performance statistic of the classifier will be averaged over all the results.

3. Exploratory Data Analysis

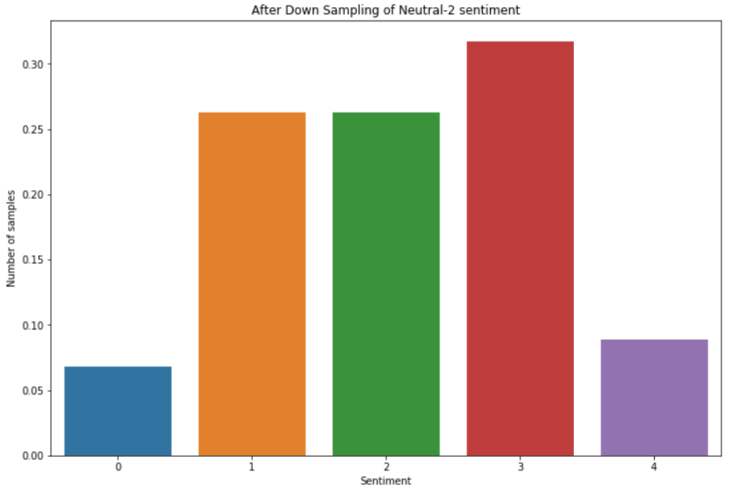
We need to analyse and pre-process the data in order to prepare it for running classification models. This comes under exploratory data analysis.

3.1 Handling the Imbalanced Data

Firstly, on analysing the given data, we found that the train data is biased towards sentiment 2 i.e. neutral(Refer histogram below) since almost 50 percent of the whole data evaluates to ‘neutral’.



Therefore, we downsampled the majority class in order to achieve an almost equal distribution across all sentiments.



We kept two copies of data-the original data and the down-sampled data. All the proceeding steps were performed on both sets of the data so that we can compare the results and choose the model that gives us the best prediction accuracy.

3.2 Data Cleaning

Furthermore, in order to gain better insight into the training data, we performed the following steps to analyze the data:

1. Calculated the average count of phrases per sentence in the given train set which came out to be 18.2975.

2. Found out the number of phrases in train set- 156060

3. Found out the number of sentences in train set- 8529

4. Calculated the average word length of phrases in train- 7.19

5. Extracted 30 most common unigrams, bigrams and trigrams in each sentiment.

We notice that all of them contain a lot of stop words and punctuations which may lead to predicting the false sentiment for the review since similar words have high counts in more than one sentiment. Therefore, we need to clean the data in order to get better results. In order to resolve these issues, we performed the following pre-processing steps on the data:

1. Lemmatization- Firstly, we grouped together all inflexions of a word in order to analyse them as a single item during class prediction.

2. Changed the entire corpus to lowercase in order to make our predictions case-insensitive by not differentiating between the same word in different cases.

3. Removed all stop words like prepositions, articles, conjunctions etc. since they do not add much value to our predictions and are fairly common to all reviews

4. Used Regex formatting to remove punctuation marks like exclamation, full-stop, comma etc. since they are useless towards predicting the class label.

5. Breaking the grammatical structure of the data in order to get a corpus of words that can be used as our feature set.

6. Removed null values encountered due to data cleaning(removal of stop words, punctuations etc).

After completing the data cleaning, we again tried to extract the 30 most common unigrams, bigrams and trigrams for each sentiment. We found a much better concise and useful list of words this time.

3.3 Word Cloud

The sentiment of a review can be predicted by exploring the words that are most likely to draw the review towards a particular sentiment. To analyze this, we can visualize the data by creating a word cloud using the word cloud library in python. It will give us some insight into the words and their frequencies in a review for a particular sentiment. We created the word clouds for the clean down-sampled text data in order to avoid all stop words and focus on the unique root words.

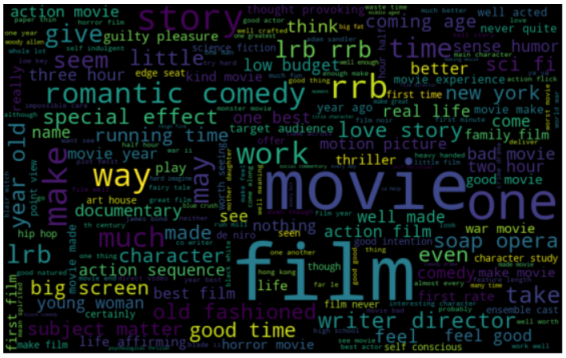


Figure 1. All words

Figure 2. Positive Sentiment

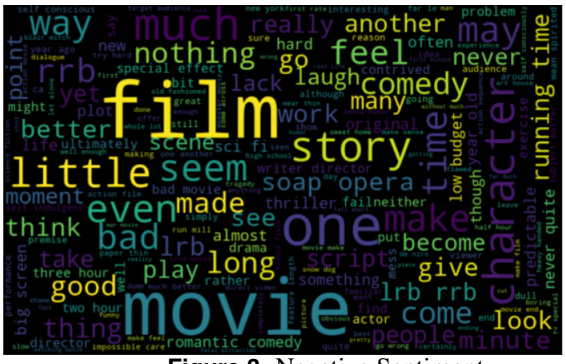


Figure 3. Negative Sentiment

3.4 Splitting the data

Being a Kaggle contest, we have been provided two files: a train file will specified class labels and a test file for which we need to predict the sentiment using our model. Since the test data is not labelled, we split our train file into 80:20 ratio to create a validation file. We will be using this validation file to validate the predictions using our model before running it on the test data.

4. Feature Extraction

For extracting meaningful reviews in order to train our classifier models, we used the following techniques:

4.1 Bag of Words

This is a text mining technique that builds a predefined set of words and calculates the word count for each word. This word count matrix is then used to extract different feature representations. We have a huge amount of data owing to which the number of words in the review text will be huge. Therefore, in order to get better results, we will consider the 10 top most frequent words as our feature set and train our models on these features.

4.2 N-grams

We tried using N-grams to create unigrams and bigrams. This method provides more contextual information on the review text. It combines n sequential words. E.g. we have a review which states "Movie was not good". It clearly implies a negative sentiment. However, if we take each word separately, we will not detect this. N-grams will be most probably be able to learn that "not good" is a negative sentiment. However, when we tried to use N-grams in this context, for 2 reviews, 30 features were getting created. Therefore, as expected, we encountered a memory error when we tried to run it across the entire review dataset, thereby eliminating n-grams feature extraction method.

4.3 Tf-Idf (Term Frequency Inverse Document Frequency)

Tf-Idf (Term Frequency Inverse Document Frequency) It is a text mining technique used to categorize documents. It emphasizes on words that occur frequently in a particular review, while at the same time de-emphasizing words that occur frequently in every review. This method will help us concentrate on portions which might be less frequent but have more significance for the overall polarity of the review. To limit the number of features getting created, we put the max_features parameter as 60,000 for this dataset.

5. Baseline Models

The basic aim of this project is to test different models to classify the movie reviews into Positive, Somewhat Positive, Neutral, Somewhat Negative and Negative classes. To achieve this, we started with testing several baseline multiclass classification algorithms on the bag-of-words model to find out which model is working better. Also, we need to test whether we are getting better accuracies for the original dataset or the downsampled dataset. The performance evaluation is done on the validation set. We used Sklearn implementation of the classifiers for this project.

5.1 Naive Bayes

It works on the principle of Bayes' theorem with the assumption of conditional independence between every feature(word or group of words). We used Multinomial Naive Bayes since we have 5 classes in total.
Accuracy on Downsampled data: 0.3223524852243
Accuracy on Original data: 0.5010508617065994

5.2 Logistic Regression

It is a predictive classifier in which there are one or more variables determining the outcome. The probabilities describing the possible outcomes of a single trial follows a sigmoid curve.
Accuracy on Downsampled data: 0.32332138358686174
Accuracy on Original data: 0.5023765641672325

5.3 K-Nearest Neighbours

It is a lazy-learning classifier because it has no predefined decision boundary. The data is classified simply by majority voting between k number of nearest neighbours. We used 5 nearest neighbours to predict the classes in our case. However, this model was extremely slow and therefore eliminated from our classification problem.
Accuracy on Downsampled data: 0.2894099408971999
Accuracy on Original data: 0.48983089210075337

5.4 Support Vector Machines

It is a supervised learning algorithm which designates a hyperplane in an n-dimensional space where each data instance is plotted that efficiently differentiates each class. We used LinearSVC that finds the best fit hyperplane to categorize the data. Thereafter, the features can be fed into the model to find the class it belongs to.
Accuracy on Downsampled data: 0.32366049801375835
Accuracy on Original data: 0.5024735667863033

5.5 Summary of Baseline Models

Feature Extraction Using Bag of Words	Logistic Regression Model	Multinomial Naïve Bayes Classifier	K-NN Classifier	Linear SVM Classifier
Down-Sampled-Data	0.318	0.316	0.28	0.31
Whole Data	0.51	0.50	0.49	0.51

Figure 4. Baseline Models Accuracy Summary

After applying these baseline models, we realized that the original dataset gave us much better results in terms of accuracy than the downsampled data. Therefore, we eliminated the downsampled dataset to find the best model. Also, KNN being super-slow was eliminated in the further steps.

Also, since this project deals with sentiment analysis of textual data, we realized that bag of words concentrates more on word count rather than the logical construct of data. It is quite possible that a word may occur less frequently but is clearly indicative of a particular class. To overcome this shortcoming, we used TF-IDF model for feature extraction in the subsequent models for better accuracy.

6. Experiments and Results

For better accuracy, based on the baseline models, we extracted features using TFIDF, hypertuned the model parameters and used pipeline over these models to get better results. We started by applying hyper parameter tuning for TF-IDF vectorizer by supplying a range of maximum features(max_features) and built the pipelines for the respective models.

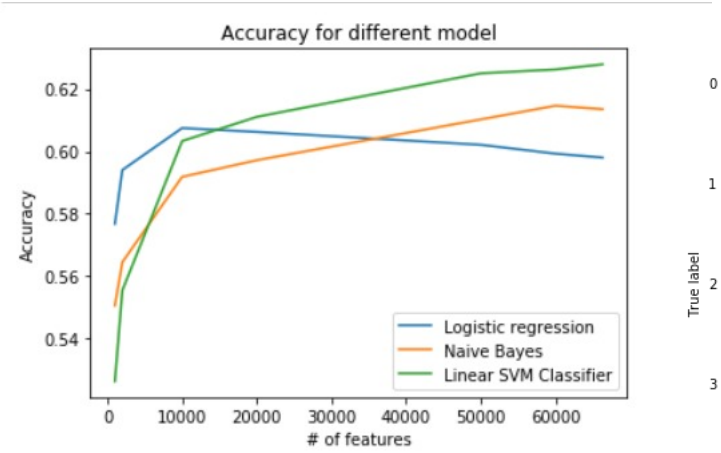


Figure 5. TFIDF Feature Tuning

6.1 Pipeline Models

Pipeline sequentially applies a list of transforms to estimate the final class.

We constructed confusion matrices summarizing the correct and incorrect predictions for each class. Models implemented:

1. Logistic Regression using TF-IDF

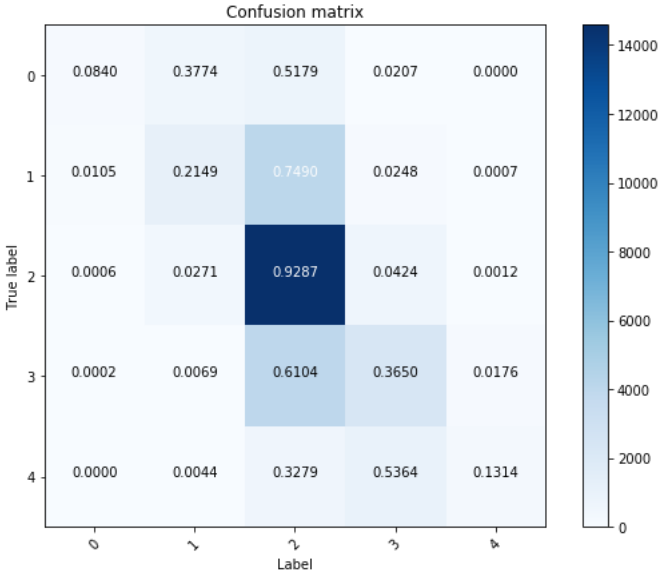


Figure 6. Logistic Regression

As per the confusion matrix for logistic regression, the model predicted label 2 well, however it did not perform well for the other sentiments. The test accuracy was also not that great for this model, therefore, we tried training our data by pipelining TFIDF with Multinomial Naive Bayes.

2. Multinomial Naive Bayes using TF-IDF

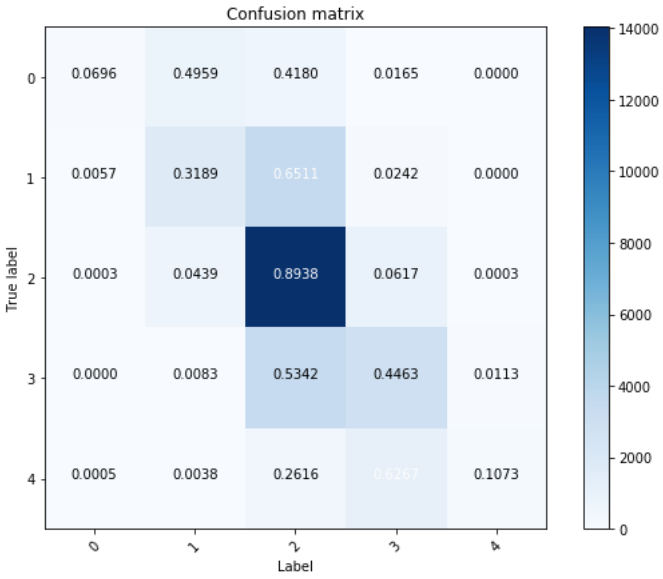


Figure 7. Multinomial Naive Bayes

Again, this model predicted the sentiment classes better as compared to logistic regression, however, we went on to experiment the learning using Linear Support Vector Model to check if it provided better results.

3. Linear Support Vector Classification using TF-IDF

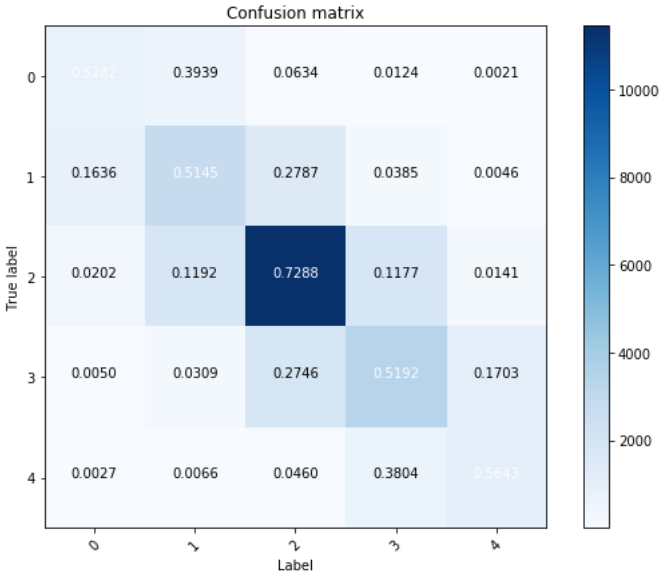


Figure 8. Linear Support Vector Classifier

By analyzing the confusion matrix for this model, we found that it performs better than the other two models.

6.2 K-Fold Cross Validation

We also implemented 5-fold cross validation wherein we divided our train data into 5 folds and treated all combinations of 4:1 wherein 4 folds will be our train data and 1 fold will be

our test. Thereafter, we found the average accuracy over each fold. K-fold cross validation results summarized below:

Models	5 Fold Cross Validation Scores
Logistic Regression	0.343304
Naive Bayes	0.358515
Linear SVC	0.535127

Figure 9. 5-Fold Cross Validation

7. Summary and Conclusions

In a nutshell, the accuracy scores for each model over down-sampled as well as original data over each baseline and improvised model can be summarized as below:

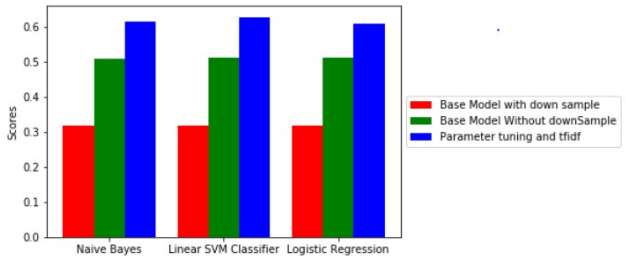
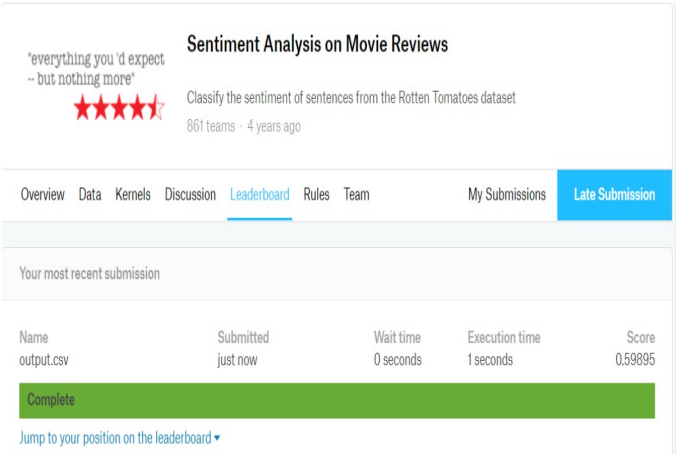


Figure 10. Model Comparison

Also, when we tried to predict the classes for the Kaggle test set provided to us, this model provided the best accuracy of 59.8 percent. Thus, we chose pipelining of TF-IDF with Linear Support Vector Classifier as our final model.



Regarding this project, our tasks of learning the challenges associated with sentiment analysis and trying to resolve them by trying different features selection and classifier models have been met. However, we could try n-grams if we had better memory and it could have provided us with better results. Additionally, trying other feature extraction techniques such as Random Forests feature importance module or neural networks like Long short-term memory models could increase the performance over sentiment analysis of movie reviews.

Acknowledgments

We have put in immense efforts in this project. However, it would not have been possible without the kind support and help of our Professor Dr Gregory J Rawlins for giving us the opportunity to work on this project. We would also like to extend our sincere gratitude to our mentor Mr Manoj Joshi and Associate Instructors Zeeshan Ali Sayyed and Christopher Torres-Lugo for their continuous help and guidance throughout the completion of this project.

References

1. https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/
2. https://www.kaggle.com/adamschroeder/countvectorizer-tfidfvectorizer-predict-comments
3. https://www.kaggle.com/ynouri/rotten-tomatoes-sentiment-analysis
4. https://www.kaggle.com/artgor/movie-review-sentiment-analysis-eda-and-models
5. https://pdfs.semanticscholar.org/799b
6. http://cs229.stanford.edu/proj2006
7. Pang, B., L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In EMNLP 2002, 79–86. http://www.cs.cornell.edu/home/llee/papers/sentiment.pdf
8. Rennie, J., Shih, S., Teevan, J., Karger, D. Tacking the Poor Assumptions of Naive Bayes Text Classifiers. In ICML-2003. http://haystack.lcs.mit.edu/papers/rennie.icml03.pdf
9. Siolas, G., d'Alche-Buc F. Support Vector Machines Based on a Semantic Kernel for Text Categorization. In IJCNN 2000. http://ieeexplore.ieee.org/iel5/6927/18674/00861458.pdf
10. Toutanova, K. "The Stanford NLP Group Part-of-Speech Tagger." 2006. http://nlp.stanford.edu/software/tagger.shtml
11. https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/
12. https://medium.com/coinmonks/solving-twitter-sentiment-analysis-problem-on-analytics-vidhya-ea3e51eea521