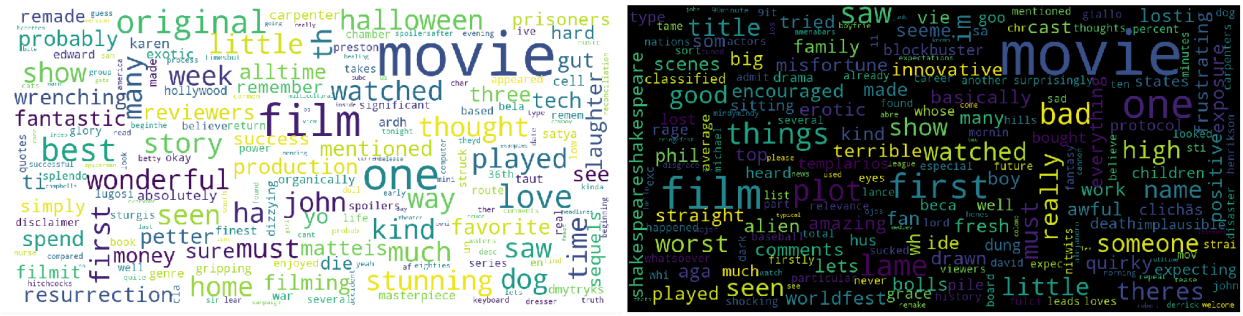
Last Modified: Dec 11, 2019

Sentiment Analysis of Movie Reviews

Mamtaj Akter
makter2@cs.uoregon.edu

Abstract- Finding out a rating of a movie before spending money to buy a movie CD or spending hours for watching that movie is significantly important for viewers who don't want to spend time and money on a dull and boring movie. Thousands of reviews on a movie can be found online, some are positive and some are negative, but the intended viewers of that movie might get overwhelmed with these multi-sentimental reviews and might have a hard time to make the right decision whether he or she should subscribe for the movie or not. In this project, I attempt to analyze the sentiment of reviews by carefully analyzing the collected data, cleaning, and enriching dataset with new insightful features. Then, I use the cleaned and enriched data to train different machine learning models and evaluate predictions. After a careful feature selection and hyper parameter tuning, I finally use Random Forest, Naïve Bayes, Logistic Regression, Support Vector Machine and Bidirectional Long Short-Term Memory (LSTM) deep learning model to get a comparative analysis. However, my results suggest that further improvement in model training requires more analysis, experiment, and comprehensive features.

I. INTRODUCTION



(a) (b)

Figure 1: Word clouds of a) positive reviews and (b) negative reviews

A movie production is usually the largest investment in the media world that a movie producer makes before making a movie. Gaining trusted insight into the previous movies of a director before making an investment or monitoring the overall sentiment of the director's previous movies is significantly important. On the other hand, the mass people who loves to spend their weekend or their free time watching a quality movie with their closed ones, conducting a little research on the movie is very important but can be too cumbersome for them. The main challenge of this movie review analysis is that there are millions of reviews or texts found on the internet. It is nearly

impossible to manually read every viewer's review and analyze those to find out the right answer of this question- is this movie really worth it to watch? So, this project attempts to predict the sentiments of those numerous reviews. And, in order to do that, I need machine learning or deep learning algorithm. Many work have been conducted on the movie review analysis. In general, the analysis on people's sentiment or feelings are called Sentiment Analysis [1]. The issue that makes text classification or sentiment analysis difficult is that even though a text can have several positive words, the overall interpretation of the text can be negative sentiment. For example, let's consider Figure 1. Although Figure 1 (a) depicts a word cloud of positive reviews where I find a lot of neutral words, for example- show, saw, three, send, etc. In Figure 1 (b) we see a few positive words too, even though it's a word cloud made from negative reviews.

Using publicly available data about the movie reviews, in this project, I attempt to predict sentiments of each reviews. And I am going to try several machine learning models and deep learning model that includes Multinomial Naïve Bayes, Random Forest, Logistic Regression and Support Vector Machine. The rest of the paper is organized as follows: Section-II provides a detailed description of the given problem, data description and the approach to solve the key issues of the given problem. Section-III highlights my experiment and the results I obtained. Next section discusses the measures I got for each model. Finally, I conclude in Section-V.

II. BACKGROUND

A. Dataset Selection

Since my focus is on analyzing movie reviews and predict the sentiment of those, I have used the IMDB dataset from [3]. This dataset has 50,000 movie reviews with only one feature "review" and the label "sentiment". This is a dataset for binary sentiment classification and half of the reviews are positive and the other half are negative. So, although this dataset is totally labeled, for this project I need to predict the number of positive and negative reviews using machine learning classifiers and deep learning algorithms.

The graph in Figure 2 illustrates the variation of the word count for each review. It clearly shows that the average number of words per review was around 230. Thus, I can assume that in general, people write pretty descriptive review.

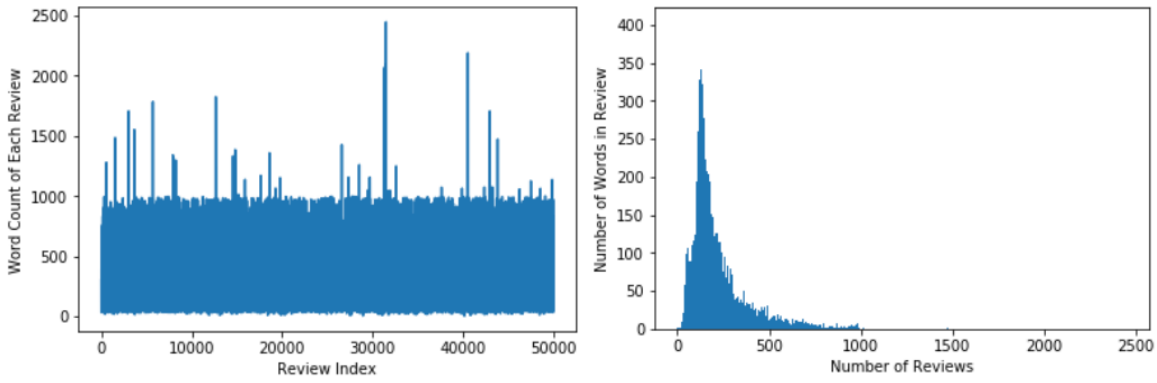


Figure 2: The Number of Words in Reviews and the Histogram

B. Data Preprocessing

The most challenging part of this project was to clean the data before applying the classifiers. The first step is to convert the labels from strings to binary values to feed to the classifiers that I am going to use, since most of the machine learning classifiers require numeric labels. Then I convert all characters in the review text to lower case. I found a lot of punctuations in each review which are not important for the analysis. The challenge that I faced in this step is removing the html tags. A typical review text looks like this:

*This German horror film has to be one of the weirdest I have seen.

I was not aware of any connection between child abuse and vampirism, but this is supposed based upon a true character.

My hero is deaf and mute as a result of repeated beatings at the hands of his father. he also has a doll fetish, but I cannot figure out where that came from. His co-workers find out and tease him terribly.

During the day a mild-manner accountant, and at night he breaks into cemeteries and funeral homes and drinks the blood of dead girls. They are all attractive, of course, else I wouldn't care about the fact that he usually tears their clothing down to the waist. He graduates eventually to actually killing, and that is what gets him caught.

Like I said, a very strange movie that is dark and very slow as Werner Pochath never talks and just spends his time drinking blood.*

I use HTML parser [8] to remove the HTML tags from the reviews. Then, I remove all punctuations using simple regular expression and tokenize the review text into single words. Text tokenizing is a way of splitting the text to tokens. A token can be either be a word, a sentence or a paragraph. In this work, I have used Python NLTK word tokenizer [4]. Next, I perform some stemming. Stemming is a process which essentially removes the morphological axes from words so that I am left with the root word. The main idea of stemming is to normalize the review texts so that they carry the same meaning regardless of the tense. In this project, I use the Porter Stemmer [5] for stemming the text. Then I also remove the stop words- most frequent words, for example: the, this, there, is, were, etc. Usually, stop words don't make any sense for sentiment analysis. So, it's important to remove such words. The NLTK module contains a list of stop words. Each word of each review is matched with the corpus to determine whether it is removed or to be kept.

C. *Feature Engineering*

Since I have only one independent feature “review” in the dataset, I did not have to do a lot of feature engineering. I transform the review texts into word-count vectors which will be the features that I will feed into my machine learning models. I then transform the word-counts to term frequency-inverse document frequency vectors [6] for the machine learning classifiers. However, I have transforms each tokenized review to a sequence of integers (each integer being the index of the token in a dictionary). For this step, I have used Text preprocessing Tokenizer [7].

D. Method Selection

The overall task in this project is to predict text classification of reviews as positive or negative. Therefore, I explored several classification machine learning and deep learning models. I used Logistic Regression, Multinomial Naïve Bayes, Random Forest, and SVM Classifier. I also

applied a deep learning neural network model like Bi-LSTM. For all of the above models, I used sklearn [9] and keras.layers [10] modules by tuning their hyper-parameters.

III. METHODS

In this section, I describe the experiments conducted and analyze the results that I have found.

A. Experimental Procedure

I take 80% of the labeled data for training. For cross validation and test, I equally (almost) divide the rest of the 20% of the data without label. So, I have 40950 training examples, 4500 reviews for the cross validation and 5000 examples for testing to experiment on.

B. Tuning Parameters

In case of hyper-parameter tuning for the deep learning, I set the hidden nodes = 32 for the Bidirectional LSTM. For the dimensionality of the output space in dense function of keras.layers I set 20. To prevent overfitting, I set the value of the very popular regularization technique dropout= 0.05.

C. Correlation of the Predictions

After training the models I get almost similar (except Random Forest Classifier) accuracy values for both the validation and test data. I also have found that the accuracy of the validation dataset is pretty similar to the accuracy percentage of the test data.

Method Name	**F1 scores on Validation Dataset**	**F1 scores on Test Dataset**
Multinomial Naïve Bayes	0.85%	0.86%
Support Vector Machine	0.89%	0.90%
Logistic Regression	0.89%	0.89%
Random Forest Classifier	0.74%	0.75%
Bi-LSTM	0.87%	0.87%

Table 1: F1 scores of Validation and Test Examples

IV. DISCUSSION

As discussed above, I tried multiple machine learning classification models on the feature representation of the textual information in the reviews. Among the ML models that I applied, Support Vector Machine and Logistic Regression model seemed to have best performance with classification accuracy around 90% and 89% respectively (As shown in Table 1). In text classification problems, since both the numbers of text/document and features/*words* are large,

SVM works best because of its linear kernel characteristics. Logistic Regression (LR) measures the relationship between the label/sentiment and one or more independent variables (i.e, tf-idf vector) by estimating probabilities using a logistic/sigmoid function and therefore we get a good accuracy on applying LR method. On the other hand, Naive Bayes has the major advantages – it does not require large dataset to perform a good prediction and that explains why we get such good accuracy- 86%. However, Random Forest classifier had the worst accuracy of around 75%. Since it has the ability to handle an extremely large number of features and there are, in this project, we have very limited features, this model could not improve the accuracy. While Bi-LSTM is the perfect fit for sentiment analysis, in this project I did not get a superior accuracy because of my not-so-large dataset. With the usual disadvantage of neural networking models, they also require much more data than traditional machine learning algorithms. I think, that explains why it could not give the highest accuracy.

V. CONCLUSION

From the results above, I can infer that for the problem statement, Logistic Regression and the Support Vector Machine Model with feature set including tf-idf vector is best. Random Forest classifier may not be a good selection as machine learning model here. Also, when dataset is not large enough, deep learning is unnecessary. I can expand this project by using more data sources that are publicly available (like the names of the reviewer, dates of comment, the names of the movie they are reviewing for, and so on), applying different paradigm of data analysis, feature engineering and visualization, I can be able to find out more concrete sentiment analysis in the future.

REFRENCES

[1] Sentiment Analysis. https://en.wikipedia.org/wiki/Sentiment_analysis
[2] Adnan Duric and Fei Song. Feature Selection for Sentiment Analysis Based on Content and Syntax Models. ACL Workshop on Computational Approaches to Subjectivity and Sentiment Analysis, 2011.
[3] IMDB Movie Reviews. https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
[4] Word_Tokenizer. https://www.nltk.org/api/nltk.tokenize.html
[5] PorterStemmer. https://www.nltk.org/api/nltk.stem.html
[6] Term Frequency-Inverse Document Frecuncy. http://www.tfidf.com/
[7] Keras.preprocessing.text Tockenizer. https://keras.io/preprocessing/text/
[8] HTML Parser. https://docs.python.org/3/library/html.parser.html
[9] Sklearn. https://scikit-learn.org/stable/
[10] Keras. Layers. https://keras.io/layers/recurrent/