Data Mining Project

report

on

Movie Review Sentiment Analysis

Submitted in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

Navneet Taunk (Roll No.: 18075041)

Naveen Kumar Mall (Roll No.: 18075071)

Pulkit Srivastav (Roll No.: 18075045)

Dvij Joshi (Roll No.: 18075026)

Under the kind guidance of

Dr. Bhaskar Biswas

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Indian Institute of Technology (BHU) Varanasi

Varanasi, Uttar Pradesh, India - 221005

Contents

Main Sections

1 Introduction

Movie review is very important way to test the performance of a movie. It provides a numerical rating to a movie as well as tells us about success of a movie. A collection of a movie review gives us a deeper insight on different aspects of the movie. A text movie review tells us about solid points of the movie and deeper analysis of a movie reviews can tell us if the movie meets the expectations of the public or not [1]. **Sentiment** is the underlying meaning or emotion behind something. A sentiment could be happiness, anger or generally it could be positive or negative sentiments.
Sentiment analysis [2] is a major subject in machine learning which aims to extract subjective information from textual data. This field is closely related to natural language processing and text mining. It is used to determine the emotion behind the reviewer on that review. Using this we can find the the state of mind of the reviewer while providing the review and will get to know his sentiments behind that review.

1.1 Project and Dataset Description

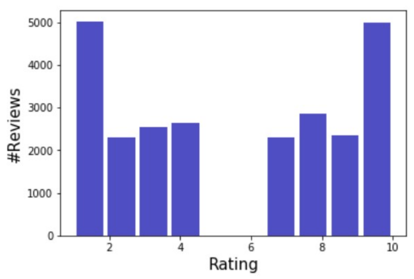
In our project, we tried to extract the subjective information from text and tried to figure out sentiment behind the text. Our aim of the project is to use sentiment analysis techniques to determine the overall reaction of the reviewer, whether or not they liked it, their emotions while writing them, etc. by examining a set of movie reviews. We used **Large Movie Review (IMDB) Dataset** [3] which was used by the AI department of stanford University for their publication [4]. The core dataset contains $50,000$ reviews which are evenly split into $25,000$ training set and $25,000$ test set. The dataset was collected from IMDb [5]. In dataset each review is labelled with the ratings of the movie on scale of 1-10. We categorized these labels into bipolar sentiment like/positive class or dislike/negative class. The overall distribution of data is balanced and train and test sets are disjoint. We classify the rating in negative/dislike(0) class if rating of the review is less than equal to 4 and if review rating is greater than equal to 7, then into a positive/like(1) class.
An example of review text is:
¡html¿[18:02 3/8/2011]:Bromwell High is a type of cartoon comedy. It broadcasted at

the same time as other programs describing school life, such as "Teachers". My 35 years in the teaching profession has led me to believe that Bromwell High's satire is very much closer to reality than is "Teachers". The scramble to survive financially, the insightful students who can see right through their pathetic teachers' pomp, the pettiness of the complete situation, all reminds me of the schools I knew and their students. When I saw the episode in which a student repeatedly tried to burn down the school, I immediately recalled .... at .... High.¡/html¿ (@172.16.254.1)

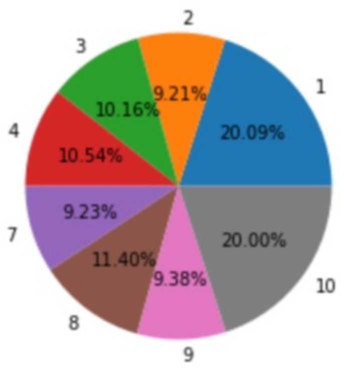
As we see that this review has lot of impurities like HTML tags, punctuations and other impurities. Therefore Dataset must be cleaned and pre-processed before training. In the upcoming section, we shall clean the dataset and will visualize our dataset.

2 Data Visualization and Data cleaning

We have 25,000 training data and 25,000 test data and each review was text file with following naming convention as *reviewid$_r$ating.txt*. We made a new vector y which is contain ratings of each review. We plotted these data using matplotlib to visualize it.



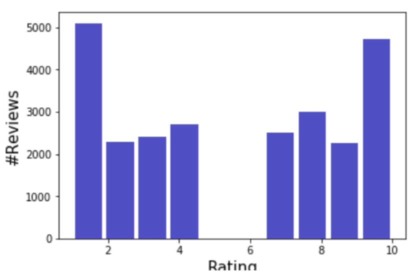
(a) No. of reviews vs Rating for training Data



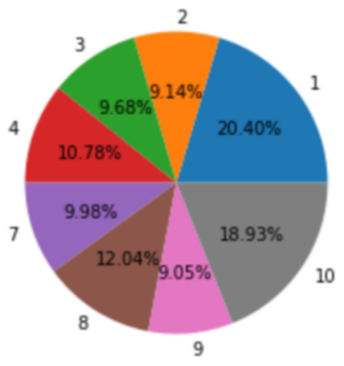
(b) Pie Plot of training data

Figure 1: Data visualization for training set

We can clearly see that data is balanced that is we have 12,500 positive class training examples and 12,500 negative class training examples. The above two plots



(a) No. of reviews vs Rating for Test Data



(b) Pie Plot of training data

Figure 2: Data visualization for Test set

visualize our datasets clearly, but in reviews we have many impurities that must be cleaned before data will be send for pre-processing and further training. A typical examples of reviews from our dataset is as follows:

Bromwell High is a type of cartoon comedy. It broadcasted at the same time as other programs describing school life, such as "Teachers". My 35 years in the teaching profession has led me to believe that Bromwell High's satire is very much closer to reality than is "Teachers". The scramble to survive financially, the insightful students who can see right through their pathetic teachers' pomp, the pettiness of the complete situation, all reminds me of the schools I knew and their students. When I saw the episode in which a student tries repeatedly to burn down the school, I immediately recalled .... at ..... High. A classic line: INSPECTOR: I'm here to sack one among your teachers. STUDENT: Welcome to Bromwell High. I expect that many adults of my age think that Bromwell High is far-fetched. What a pity that it isn't!

We can clearly see that this dataset is noisy because there is no fixed standard of writing reviews. Therefore, Dataset must be cleaned before further steps to obtain meaningful results. Removal of alphanumeric characters i.e. ip address, time, date and punctuations is **Data Cleaning**.

We converted everything to lowercase letters and removed non-english and meaningless words. We use **Beautiful Soup** to remove HTML tags from the text and removed all punctuation marks such as '/', ';' etc as they don't provide any substantial information.

3 Data Preprocessing

We get the cleaned data after cleaning it but still we have to do some preprocessing to make it useful. Conversion of cleaned data to machine readable format is termed as **Data Preprocessing**. Data Preprocessing involves two steps:

1. Stop Words Removal

2. Stemming

3. Bag of Words

3.1 Stop Words Removal

A stop word is a commonly used word such as 'is', 'am', 'the', 'a', 'an', that add no meaning to our data. These are meaningless word that must be removed. We don't want these words in our dataset because it will take unnecessary space and increase the processing time. We removed these words by storing a list of words that are considered to be stop words. For this we used **NLTK** [7] library. Here are few words which are stop words.

'being', 'their', 'to', 'having', "wasn't", 'into', 'her', "mustn't", 'yourself', 'hadn', 'where', "don't", 'very', 'most', 'while', 'such',..........'if', 'wasn', 'below', 'himself', 'when', 'our', 'until', "you've", 'now', 'more', 'should'.

We removed such words from our dataset.

Text before removing Stop words	After removing Stop words
bromwell High is a cartoon comedy. It ran at the same time as some other programs about school life such as teachers	bromwel high cartoon comedi ran time program school life teacher

Table 1: Stop Words Removal

3.2 Stemming

Stemming is a process of converting inflected words to its root word. For example words like likes, liked, likely will change to its root form like. We used porterStemmer in our preprocessing. We import PorterStemmer from nltk.stem.porter library. The idea behind this is that suffixes in the English language are composed of a group of smaller suffixes. This Stemmer is known for its pace and simplicity. The following table shows few examples in which each word has been changed to its root form.

Inflected Words	Root Words
teachers	teacher
teaching	teach
saw	see
fetched	fetch

Table 2: Stemming

3.3 Feature Extraction

We used CountVectorizer from scikit-learn library in python for extracting features from text data. It has been utilised to transform a given text into a vector on the basis of the count of each words that is occuring in the whole corpus. Before that we plotted Review length(word count) vs No. of reviews to get better insight of reviews. The spread is similar in shape for both types of reviews however negative reviews are on average a tad shorter.

CountVectorizer generates a matrix for some text in which every unique word is represented by a matrix column, and each row matrix represents a sample from review. The value of each cell is the frequency of the word in that particular sample itself.

4 Model Training

The overall task of this project is for the classification of reviews into two classes i.e. positive and negative. Therefore for classification we used various classification models. We used models like Multinomial Naive Bayes's classifier as this is primarily used in text mining, then we used SVM classifier and LSTM classifier with embeddings and without embeddings. For all these models we used **sklearn** [8] modules by tuning there parameters.

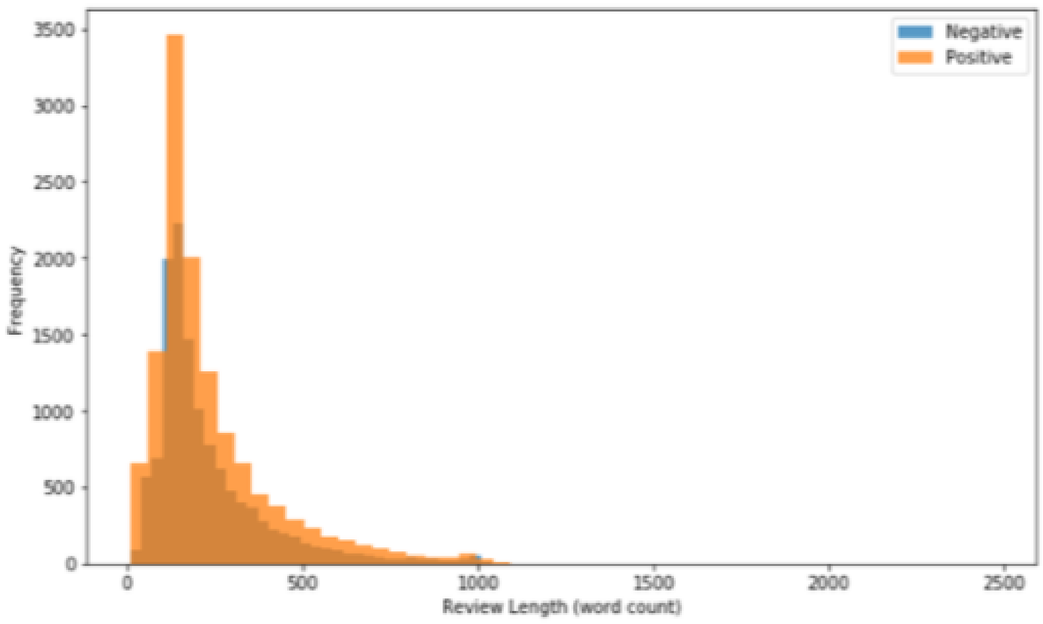


Figure 3: Word Count vs No. of reviews

4.1 Multinomial Naive Bayes

Multinomaial Naive Bayes implements the naive Bayes algorithm for multinomial data that is distributed, and is among the two classic naive Bayes variants used in classification of text where representation of the data is typically done in the form of word vector counts [9].

Naive Bayes

Naive Bayes methods are supervised machine learning algorithms that are based on Bayes theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n,:

$$P(y|x_1, ..., x_n) = P(y)P(x_1, ..., x_n|y)/P(x_1, ..., x_n)$$

and using the naive conditional independence this relationship is simplified to :

$$P(y|x_1, ..., x_n) = P(y)\prod_{i=1}^{i=n} P(x_i|y)$$

We used

$$classsklearn.naive_bayes.MultinomialNB(*, alpha = 1.0, fit_prior = True, class_prior = None)$$

for our model. We got an accuracy of 0.8524 on the test dataset of $25,000$ entries. Results were quite balanced for both the positive and negative sentiments.

4.2 Support Vector Machine

We have used Support Vector Machine model to fit our training data and make predictions on the basis of it. Support Vector Machine is a machine learning algorithm which can be used to solve classification or regression problems. It separates the data points and finds an optimal boundary between the possible outputs [6]. In classification problems, SVM separates the data points into different classes using an optimal hyperplane.
Consider an example of binary classification, where data points are to be separated into two classes. Consider a hyperplane H and let $d+$ be the minimum distance between the positive data points and H, and $d-$ be the minimum distance between negative data points and H. Define margin as the sum of $d+$ and $d-$. The best optimized hyperplane would the one with the maximum margin and it is known as the maximum marginal hyperplane. The task is to solve the optimization problem and find out the maximum marginal hyperplane. SVM algorithm uses a kernal trick to find a hyperplane that is best fitted according to the given data. We have used a linear kernel to fit the data.
In this project, we have used $svm.SVC(kernel =' linear')$ method defined in the sklearn library. We have obtained an accuracy of 0.86172.

4.3 LSTM

A **Long Short-Term Memory Network**, commonly referred to as LSTM Network, is a type or rather extension of Recurrent Neural Networks (RNNs). LSTMs are designed in way that enables them to perform well in processing, learning and classifying sequential data. Areas of application are *Sentiment Analysis, Modeling of Languages, Video Analysis and Speech Recognition to name a few.*

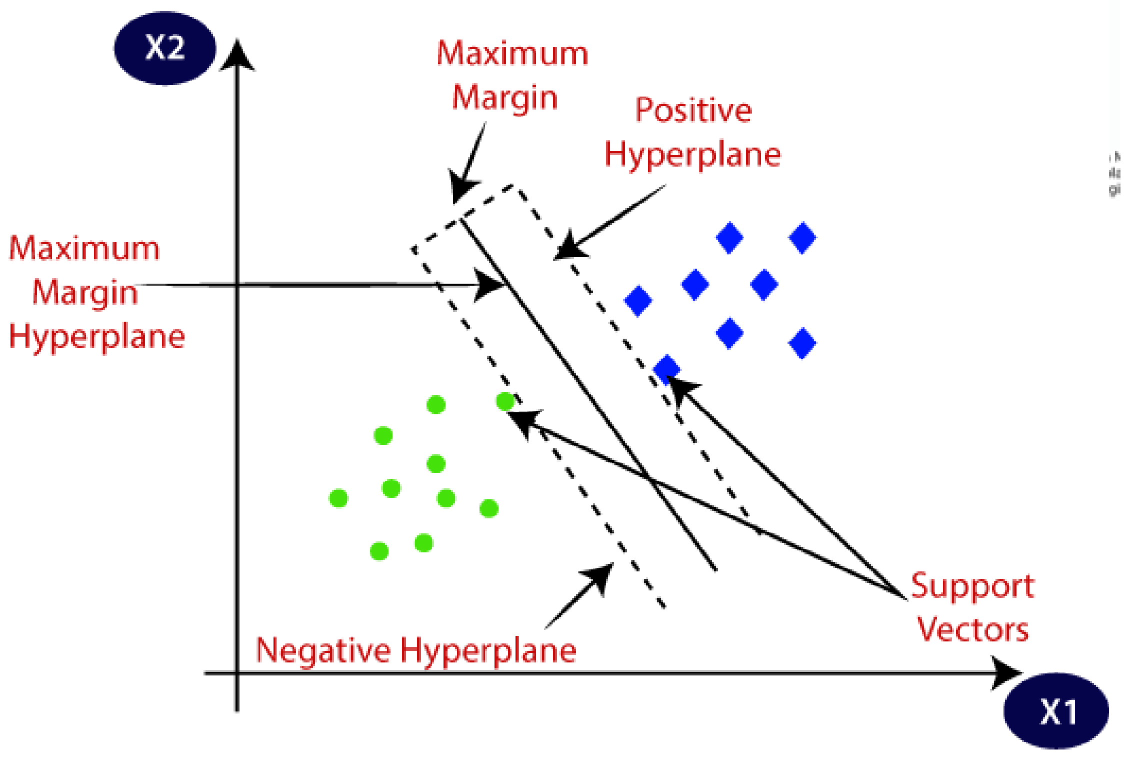


Figure 4: Pictorial Description of SVM

The standard and most popular way to train a Recurrent Neural Network is by **Back-propagation** through time.The issue of the vanishing gradients, however also causes the parameters to catch short-term dependencies while the data from previous time steps decays. There may also be the reverse problem, exploding gradients, causing the error to expand drastically with each step in time. [10]

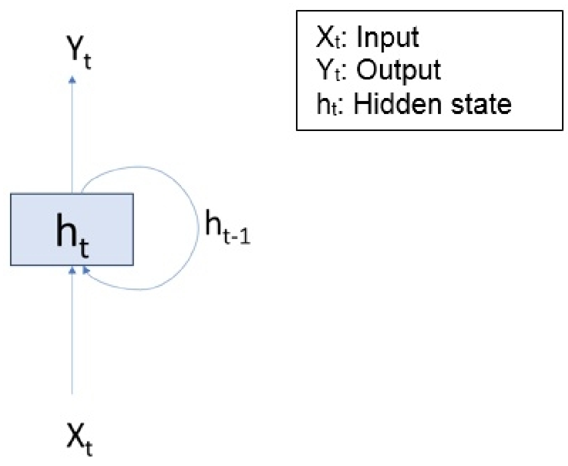


Figure 5: Recurrent Neural Network Basic Architecture

In order to overcome the underlying issue of the *vanishing gradient* in RNNs, **Long short-term memory** (LSTM) networks utilize different gates to selectively retain critical information that may be relevant and give up/forget the one that is irrelevant to the specific requirements of the application. LSTM Networks are hence better for analysis of sequential data as compared to simple RNNs, due to their lower sensitivity to the time gap.

The general architecture of an LSTM block can be visualised through the figure given below. An LSTM block typically comprises of a memory cell, input gate, output gate, and a forget gate apart from the hidden state in traditional RNNs.

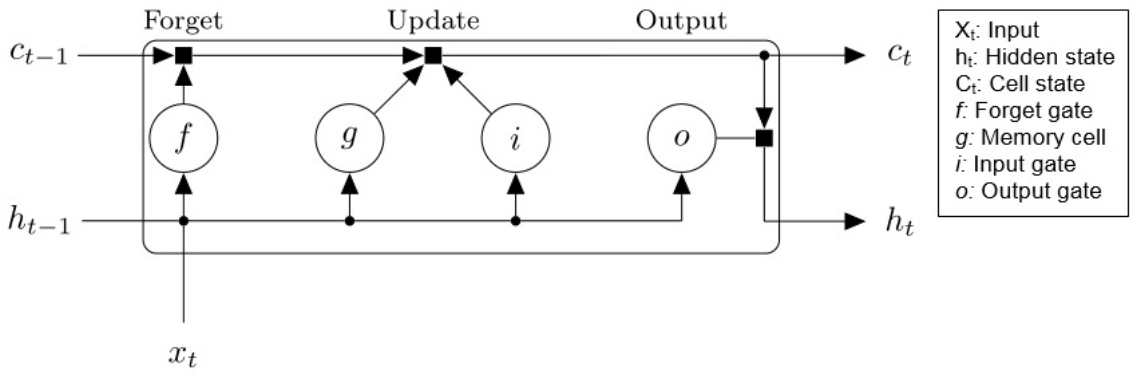


Figure 6: Long Short-Term Memory Networks General Architecture

Bi-directional Long Short-Term Memory(LSTM) Networks

Bi-directional Long Short-Term Memory(Bi-LSTM) couples two LSTM such that one LSTM is trained on sequence in direction of increasing index and the other is trained in the direction of decreasing index. Using the hidden states of both LSTM, Bi-LSTM uses both past and future context to give the output.

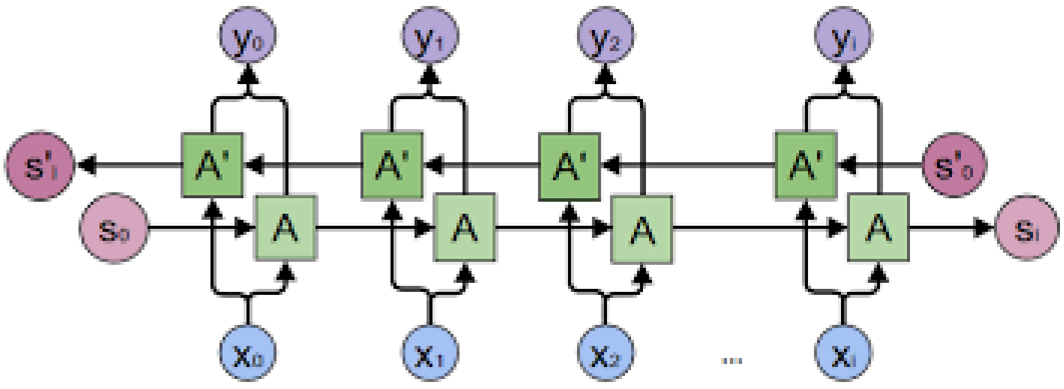


Figure 7: Bi-directional LSTM

We have implemented LSTM in Keras for analysis of our data. The architecture of our model has been depicted below:

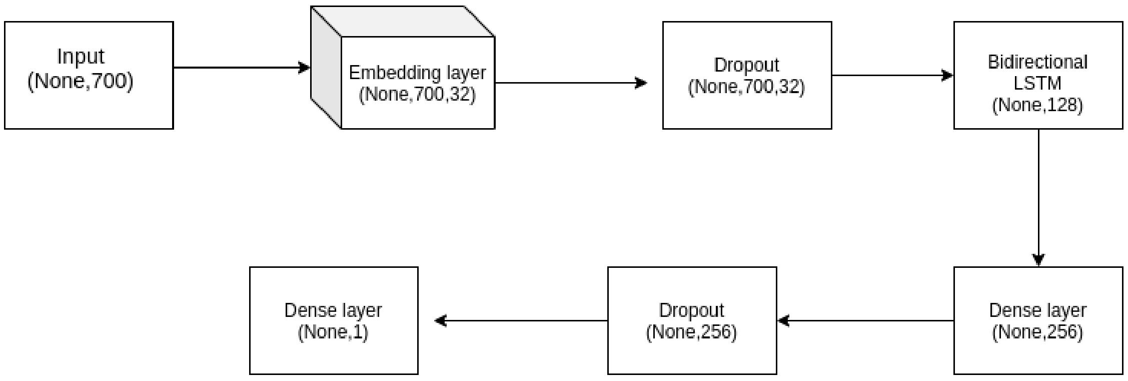


Figure 8: LSTM Model Architecture

4.4 LSTM with pre-trained word embeddings

Word Embedding

Word embedding is a feature extraction method used to represent the words of document vocabulary as a real-valued vector. Word embeddings can capture the context of a word in a document - its semantics , syntax and its relation with other words in the context . Word embeddings represent a particular word in vector form. There are various methods to generate them. One of them is **Word2Vec**. It is technique in NLP and uses a neural network model to learn word associations from a corpus of text.

GloVe Embeddings

GloVe stands for "Global Vectors". Just as Word2vec, GloVe is also a word vector technique. The significant advantage of GloVe is that GloVe does not depend just on local statistics, but also incorporates global statistics (word co-occurrence) to obtain word vectors. It thus come up with a principled loss function that uses both of the local and the global statistics. [11] We have used pre-trained GloVe embeddings in our LSTM model to obtain an accuracy of 0.8928.

FastText Embeddings

fastText is a designed library for very efficient learning of word representations as well as sentence classifications. Written in C++, it supports multiprocessing during training of data. It allows training of both supervised as well as unsupervised representations of sentences along with words. These representations (embeddings) have been be utilized for wideranged applications ranging from data compression, for selection of candidates, and even as transfer learning initializers. [12]
FastText is different from standard embeddings like word vectors (words2vec) in the sense that the latter treats every single word as the smallest unit whose vector representation is to be found but the former assumes a word to be formed by a n-grams of character, for instance, sunny is composed of [sun, sunn,sunny],[sunny,unny,nny], etc. where n could be anything from 1 to the length of the word. This way of representation of word by fastText provides multiple benefits of fastText over glove and word2vec embeddings. [13]

5 Results

Models	**Accuracy(%)**
Multinomial Naive Bayes	85.22
SVM classifier	86.17
LSTM without embedding	89.23
LSTM with Word2Vec	86.50
LSTM with Gloves embedding	89.28
LSTM with Fasttext	87.66

Table 3: Stemming

References

[1] Ankit Goyal and Amey Parulekar https://cseweb.ucsd.edu/classes/wi15/cse255-a/reports/fa15/003.pdf

[2] Sentiment Analysis- Wikepedia $https : //en.wikipedia.org/wiki/Sentiment_analysis$

[3] Large Movie Review Dataset – $http : //ai.stanford.edu/\ amaas/data/sentiment/$

[4] Andrew L Mass, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng and Christopher Potts (2011). Learning Word Vectors for Sentiment Analysis

[5] Internet Movie Database – $http : //www.imdb.com/$

[6] Support Vector Machines - $https : //www.kdnuggets.com/2017/02/yhat - support - vector - machine.html$

[7] NLTK Library- $https : //www.nltk.org/$

[8] Scikit-learn API Reference: $http : //scikit - learn.org/stable/modules/classes.html$

[9] Scikit-learn API Reference: $https : //scikit - learn.org/stable/modules/naive_bayes.html$

[10] LSTM - $https : //in.mathworks.com/discovery/lstm.html$

[11] GloVe - $https : //towardsdatascience.com/light - on - math - ml - intuitive - guide - to - understanding - glove - embeddings - b13b4f19c010$

[12] FastText Embeddings - $https : //towardsdatascience.com/fasttext - under - the - hood - 11efc57b2b3$

[13] How FastText Embeddings Differ? - $https : //www.analyticsvidhya.com/blog/2017/07/word - representations - text - classification - using - fasttext - nlp - facebook/$