Sentiment Analysis of Tweets

Using NN, Random Forrest and Transfer Learning

Aman Srivastava
B. Tech CSE
(2nd Year)
International Institute of Information Technology

Naya Raipur, India
aman13799@gmail.com

***Abstract*——— Sentiment classification is a way to analyze the subjective information in the text and then mine the opinion. Sentiment analysis is the procedure by which information is extracted from the opinions, appraisals and emotions of people regarding to entities, events and their attributes. In decision making, the opinions of others have a significant effect on customers ease, making choices with regards to online shopping, choosing events, products, entities. The approaches of text sentiment analysis typically work at a particular level like phrase, sentence or document level. In this project, we look at different approaches we can take in tackling this problem and try to build a robust sentiment analysis model which can classify tweets into positive, negative and neutral class. The model which performed best on our dataset was a fine-tuned BERT model.**

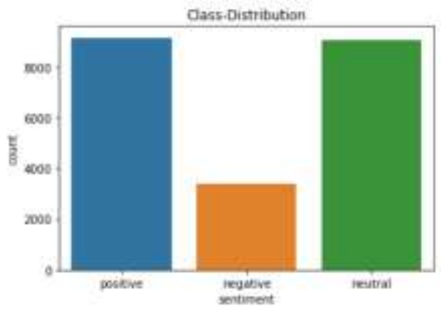
Keywords—NLP, Sentiment Analysis, Word2Vec, LSTM, RNN, BERT, Transformer, transfer learning)

I. Introduction

As internet is growing bigger, its horizons are becoming wider. Social Media and Micro blogging platforms like Facebook, Twitter, Tumblr dominate in spreading encapsulated news and trending topics across the globe at a rapid pace. A topic becomes trending if more and more users are contributing their opinion and judgements, thereby making it a valuable source of online perception. These topics generally intended to spread awareness or to promote public figures, political campaigns during elections, product endorsements and entertainment like movies, award shows. Large organizations and firms take advantage of people's feedback to improve their products and services which further help in enhancing marketing strategies. There is a huge potential of discovering and analyzing interesting patterns from the infinite social media data for business-driven applications. Sentiment analysis is the prediction of emotions in a word, sentence or corpus of documents. It is intended to serve as an application to understand the attitudes, opinions and emotions expressed within an online mention. The intention is to gain an overview of the wider public opinion behind certain topics. Precisely, it is a paradigm of categorizing conversations into positive, negative or neutral labels. Many people use social media sites for networking with other people and to stay up-to-date with news and current events. These sites (Twitter, Facebook, Instagram, google+) offer a platform to people to voice their opinions. For example, people quickly post their reviews online as soon as they watch a movie and then start a series of comments to discuss about the acting skills depicted in the movie. This kind of information forms a basis for people to evaluate, rate about the performance of not only any movie but about other products and to know about whether it will be a success or not. This type of vast information on these sites can used for marketing and social studies. Therefore, sentiment analysis has wide applications and include emotion mining, polarity, classification and influence analysis. Twitter is an online networking site driven by tweets which are 256 character limited messages. Thus, the character limit enforces the use of hashtags for text classification. There are 330 million monthly active users and 145 million daily active users on Twitter. The users tweet huge corpuses of unstructured data. These streams of tweets are generally noisy reflecting multi topic, changing attitudes information in unfiltered and unstructured format. Twitter sentiment analysis involves the use of natural language processing to extract, identify to characterize the sentiment content. Sentiment Analysis is often carried out at two levels 1) coarse level and 2) fine level. In coarse level, the analysis of entire documents is done while in fine level, the analysis of attributes is done. The sentiments present in the text are of two types: Direct and Comparative. In comparative sentiments, the comparison of objects in the same sentence is involved while in direct sentiments, objects are independent of one another in the same sentence. However, doing the analysis of tweets expressed is not an easy job. A lot of challenges are involved in terms of tonality, polarity, lexicon and grammar of the tweets. They tend to be highly unstructured and non-grammatical. It gets difficult to interpret their meaning. Moreover, extensive usage of slang words, acronyms and out of vocabulary words are quite common while tweeting online. The categorization of such words per polarity gets tough for natural processors involved. The word limit limitation of twitter forces people to use somewhat cryptic words, which needs to be preprocessed before building a model on it.

Dataset:

The dataset used for in this report was taken from Kaggle.com, provided by the inClass Competition held by IIT Kanpur in the month of May-July 2020, The Dataset contained:



1. 21630 Tweets in Train data with:
    1. positive => 9155
    2. neutral => 9075
    3. negative => 3400
2. 5398 Tweets in Test data for competition

II. Sentiment Analysis System

Sentiment Analysis is the process of finding the opinion of user about some topic or the text in consideration, it determines whether a piece of writing is positive or

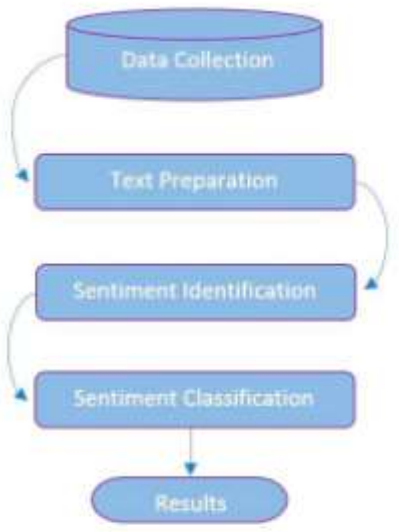


Fig. 1. Sentiment analysis model

negative[1]. The various challenges in sentiment analysis is one that the public don't always express sentiments in same way means some express in the form of ratings and some in the form of comments and second involving sentences that don't express any sentiment. The text preparation step performs required text pre-processing and cleaning on the dataset which including removal of stop words. Sentiment identification step determines the sentiment of people expressed in the text and analyzes it. Finally, sentiment classification is conducted to get the results their ideas with the others.

The sentiment analysis found in the form of comments, reviews and feedback and provides necessary information for various purposes. These opinions or sentiments can be divided into two categories: positive negative and neutral; or also categories of different rating points (e.g. 3 stars, 4 stars and 5 stars). The polarity of sentiments like "good" and "bad" also identify the sentiments either positive or negative. Sentiment analysis is the part of the text mining that attempts to define the opinions, feelings and attitudes present in a text or a set of text. It is particularly used in marketing to analyze for example the comments of the Net surfers or the comparatives and tests of the bloggers. It requires much more understanding of the language than text analysis and subject classification. Indeed, if the simplest algorithms consider only the statistics of frequency of occurrence of the words, it is usually insufficient to define the dominant opinion in a document. It is the process of determining the contextual polarity of the text, that is, whether a text is positive or negative. The use of this analysis helps researchers and decision-makers better understand opinions and client satisfaction using sentiment classification techniques in order to automatically collect different perspectives on from various platforms. There has been a large amount of research in sentiment classification. Recently, sentiment analysis has attracted an increasing interest. It is a hard challenge for language technologies, and achieving good results is much more difficult than some people think. The task of automatically classifying a text written in a natural language into a positive or negative feeling, opinion or subjectivity (Pang and Lee, 2008), is sometimes so complicated that even different human annotators disagree on the classification to be assigned to a given text [5]. Personal interpretation by an individual is different from others, and this is also affected by cultural factors and each person's experience. And the shorter the text, and the worse written, the more difficult the task becomes, as in the case of messages on social networks. In feature extraction, a sentence or document is broken into words to build up the feature matrix. In the matrix, each sentence or document is a row and each word form a feature as a column, and the value is the frequency count of the word in the sentence or document. Feature matrix is then passed to each classifier and their performance is evaluated. In this work, we have studied the classification of sentiment using 4 popular algorithms, namely Support Vector Machine, Random Forest, RNN and finally transfer learning. Text classification play an important role in many applications, it assigns one or more classes to a document according to their content. Classes are selected from a previously established taxonomy (a hierarchy of categories or classes). The text classification supports a variety of text classification scenarios like:
• Binary classification like simple sentiment analysis (positive, negative)
• Multiple class classification like selecting one category among several alternatives.
Most partitioning algorithms do not take raw text as input but numeric vectors. For this it is necessary to find a representative transformation that converts the text to digital vectors. A family of this transformation is called Bag-of-Words (BOW).

III. PREPROCESSING TWEETS

Before building any sentiment analysis model, we need to first preprocess the tweets. We need to do specific tweet text cleaning along with normal text pre-processing[2]. A tweet may contain

- URL's
- Mentions
- Hashtags
- Emojis
- Smileys
- Specific words etc.

We wrote a simple python function which takes a tweet and returns cleaned tweet. The steps involved were:

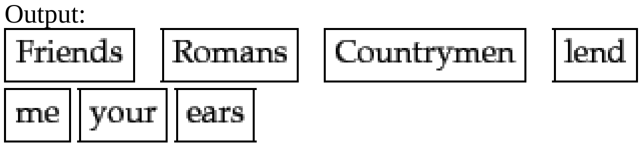
a) Removing emoji patters, (replace happy Unicode emojis with "happy" and sad Unicode emojis with "sad)
b) Removing symbols and other neutral emojis through their Unicode as they are not useful for our model
c) Replace 😊happy smileys with "happy" and ☹ sad smileys with "sad".
d) Replace English language contractions like "shouldn't" with "should not" with the help of a dictionary.
e) Removing single letter words from tweets as they are not useful in decoding sentiments of a tweet and are probably typos.
f) Removing links(URL's/Links)
g) Removing usernames and Retweet(RT) – noise in dataset.
h) remove numbers and punctuations

Tokenization:

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called *tokens* , perhaps at the same time throwing away certain characters, such as punctuation. Here is an example of tokenization:

Input: Friends, Romans, Countrymen, lend me your ears

Output:



Friends Romans Countrymen lend me your ears

These tokens are often loosely referred to as terms or words, but it is sometimes important to make a type/token distinction. A *token* is an instance of a sequence of characters in some document that are grouped together as a useful semantic unit for processing. A *type* is the class of all tokens containing the same character sequence. A *term* is a (perhaps normalized) type that is included in the IR system's dictionary. The set of index terms could be entirely distinct from the tokens, for instance, they could be semantic identifiers in a taxonomy, but in practice in modern IR systems they are strongly related to the tokens in the document.

For RandomForrest, SVM and RNN model, we also removed stop words for the English language as provided in the nltk library in python. Removing stop words made it easier for our model to generate Word2vec matrices and/or BagOfWords for building better models. Stop words are words in English language which are used to form complete meaningful sentences but are not necessary in conveying the message.

After cleaning and tokenizing the words, the train data needs to be padded as the tweets vary in length, but model has a fixed input size. We found that in our dataset the maximum length of words was 23 (words). So, we padded every tweet with blank spaces to make it of length 23 words. For our transfer learning model which takes tweet length as character, we set the max length to 256 as it's the maximum size of tweet allowed.

TF-IDF

As defined, TF is the term frequency in a single document. Terms can be words, phrases. For documents, the frequency for each term may vary greatly. Therefore, frequency is an important attribute of the term to discriminate itself from other terms. Sometimes, term frequency is directly used as the value of TF. That is, the TF value of term i is

$$TFi = tfik$$

where tfi denotes the frequency of term i in document j. Since the number of term frequency may be very large, the following formula is also often used to calculate TF value.

$$TFi = log2\ (tfij).$$

As for IDF, various formulas have been proposed. A basic formula was given by Robertson [12]. A later discussion between Spärck Jones[13] and Robertson resulted in the following formula of IDF:

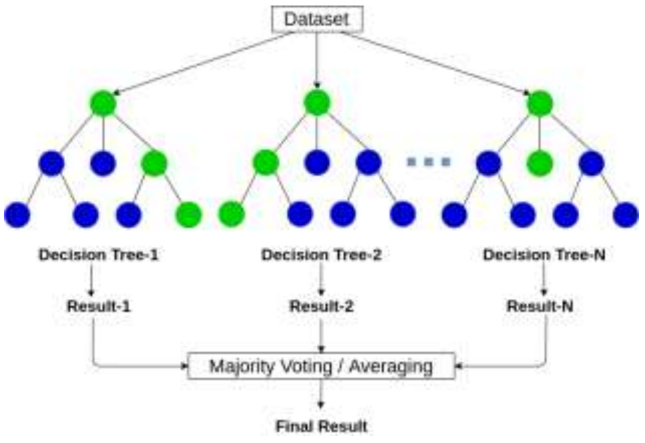
$$IDFi = log2\ (\ N\ nj\ ) + 1 = log2\ (N) - log2(nj) + 1$$

where N is the total number of documents in the collection and nj is the number of documents that contain at least one occurrence of the term i.

IV. MODELS

A. *RandomForrest*

Ensemble classification methods are learning algorithms that construct a set of classifiers instead of one classifier, and then classify new data points by taking a vote of their predictions. The most commonly used ensemble classifiers are Bagging, Boosting and Random Forest (RF). Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or the same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithms of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". [3]The random forest algorithm can be used for both regression and classification tasks. RF classifier can be described as the collection of tree structured classifiers. It is an advanced version of Bagging such that randomness is added to it. Instead of splitting each node using the best split among all variables, RF splits each node using the best among a subset of predictors randomly chosen at that node. A new training data set is created from the original data set with replacement. Then, a tree is grown using random feature selection. Grown trees are not pruned . This strategy makes RF unexcelled accuracy . RF is also very fast, it is robust against overfitting, and it is possible to form as many trees as the user wants.



The random forests algorithm (for both classification and regression) is as follows :

1. Draw $ntree$ bootstrap samples from the original data

2. For each of the bootstrap samples, grow an unpruned classification or regression tree, with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample $mtry$ of the predictors and choose the best split from among those variables. (Bagging can be thought of as the special case of random forests obtained when $mtry$ = p, the number of predictors.)

3. Predict new data by aggregating the predictions of the $ntree$ trees (i.e., majority votes for classification, the average for regression).

B. *Deep Neural Network*

A deep neural network is a neural network with more than two layers, some of which are hidden layers.

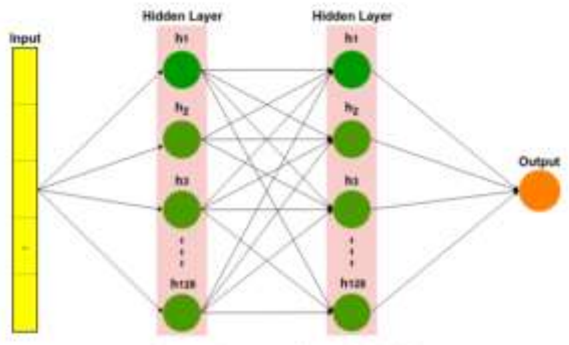
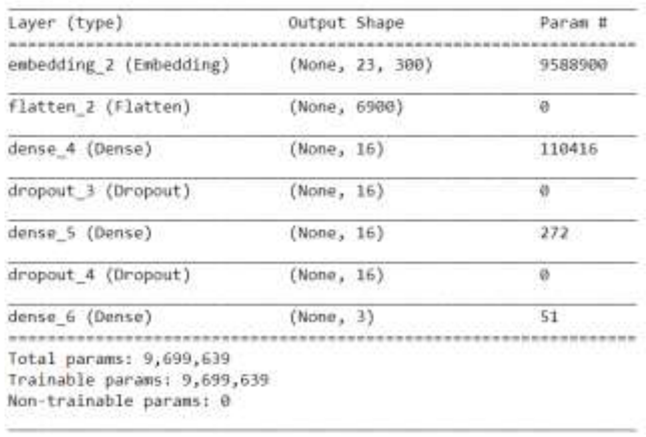


Figure 2. Deep neural network (DNN).

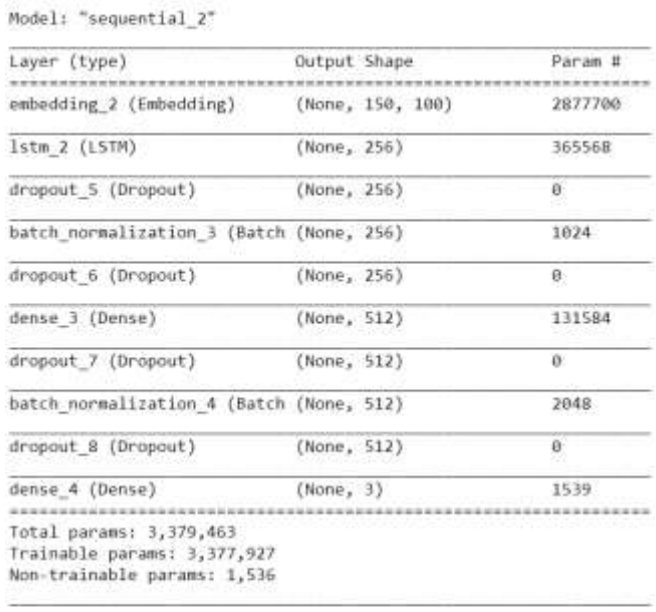
Deep neural networks use sophisticated mathematical modeling to process data in many ways. A neural network is an adjustable model of outputs as functions of inputs, which consists of several layers: an input layer, including input data; hidden layers, including processing nodes called neurons; and an output layer, including one or several neurons, whose outputs are the network outputs.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 23, 300)           9588900
_________________________________________________________________
flatten_2 (Flatten)          (None, 6900)              0
_________________________________________________________________
dense_4 (Dense)              (None, 16)                110416
_________________________________________________________________
dropout_3 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_5 (Dense)              (None, 16)                272
_________________________________________________________________
dropout_4 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_6 (Dense)              (None, 3)                 51
=================================================================
Total params: 9,699,639
Trainable params: 9,699,639
Non-trainable params: 0
_________________________________________________________________
```

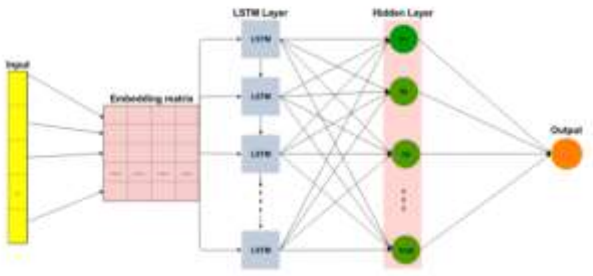
C. *LSTM*

A special type of RNN is long short-term memory (LSTM), which can use long memory as the input of activation functions in the hidden layer. This was introduced by Hochreiter and Schmidhuber (1997)[3]. Figure below



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 150, 100)          2877700
_________________________________________________________________
lstm_2 (LSTM)                (None, 256)               365568
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 1539
=================================================================
Total params: 3,379,463
Trainable params: 3,377,927
Non-trainable params: 1,536
_________________________________________________________________
```

illustrates an example of the LSTM architecture. The input data is preprocessed to reshape data for the embedding matrix (the process is like the CNN). The next layer is the LSTM. The architecture used is given below:



The final layer is a fully connected layer, which includes cells for text classification. The last layer uses the softmax activation function to reduce the vector of height 128 to an output vector of three, given that there are three classes to be predicted (positive, negative or neutral)

Transfer learning

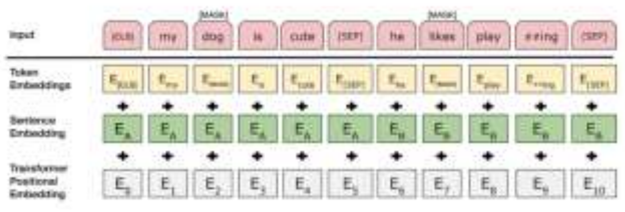
Transfer learning is a concept in deep learning where you take knowledge gained from one problem and apply it to a similar problem. While this may seem purely conceptual, it is applied quite regularly in the field of machine learning. Practically, it involves taken a pre-trained model that has already been trained on a large amount of data and then retraining the last layer on domain-specific data for the related problem. This can be a powerful method when we don't have the massive amounts of data, training time, or computational power to train a neural network from scratch. It has been used regularly in in image classification problems and in the last few years has begun to be applied in NLP. This is because of the development of BERT[5].

BERT:

BERT is a powerful model in transfer learning for several reasons. First, it is similar to OpenAI's GPT2 that is based on the transformer (an encoder combined with a decoder). However, that model can only read words uni-directionally which does not make it ideal for classification. BERT reads words in both directions(bidirectionally) and thus can read words before and after the word in a sequence. Another important advantage to BERT is that it is a masked language model that masks 15% of the tokens fed into the model. These two factors make it very good at a variety of word classification tasks.

BERT is powerful in NLP transfer learning. It is trained on a large amount of words and articles from Wikipedia. The amount of data it is trained on is much more than most people would be able to train on for specific problems. We imported a pre-trained BERT and then retrained just the final layer on sentiment analysis data to create a powerful classification neural network model in a short amount of time. Using a pre-trained BERT, we were able to achieve a F1 score of 75% without tuning many parameters.

- the pre-trained BERT model weights already encode a lot of information about our language. As a result, it takes much less time to train our fine-tuned model. We trained our model with hyper-parameter sweeps to find the best combination.
- As we are essentially performing transfer learning, we need significantly less data to build an accurate system. Compared to millions of data points required to train a model from scratch, we can accomplish the same task with only a few thousand data points.



The model hyperparameters were tweaked multiple times before reaching desired accuracy.

The hyperparameters we used:

- Batch Size: 4
- Learning rate: 1e-5
- Epochs:3

V. Discussion

Developing the project proved to be a lot more challenging than expected due to the relative inexperience we had with transfer learning. The project was competitive and kept us at our toes with over 50 teams competing to build the best model on Kaggle InClass Competition.

A. Project Limitation and challenges

With a final F1 score of .75 on test dataset, there is a large scope of improvement. The bottleneck was the presence of only 3000 negative tweets with 9000 in other two classes. There was a class imbalance. We tried data augmentation with nlpaug but there was not much improvement in test scores.

OOM error: With the BERT model being huge in size, online platforms such as Collab had CUDA OOM Error problems, so we had to train the model with reduced batch size. The accuracy might increase if trained on different hyper-parameters.

Library Dependancies: There were some initial challenges in running TensorFlow and pytorch on GPU using cuda on local devices.

B. Results:

Result Table

Model	Accuracy on Validation Set	F1 score On Test Set
Deep Neural Network	0.672	0.653
LSTM	0.678	0.69
Random Forrest	0.63	0.60
Fine-tuned-BERT	0.71	0.75

VI. Conclusion

Twitter is a source of vast unstructured and noisy data sets that can be processed to locate interesting patterns and trends. We found that for building a model, if we have less amount of dataset to build a robust model from scratch, Transfer Learning is better. We should always try transfer learning to train existing model on preprocessed data, it might reduce computations required and also get us better results.

In future work, we will focus on exploring hybrid approaches, where multiple models and techniques are combined in order to enhance the sentiment classification accuracy achieved by the individual models or techniques, as well as to reduce the computational cost. The aim is to extend the comparative study to include both new methods and new types of data. Therefore, the reliability and processing time of hybrid models will be evaluated with several types of data, such as status, comments, and news on social media. We will also intend to address the problem of aspect sentiment analysis in order to gain deeper insight into user sentiments by associating them with specific features or topics. This has great relevance for many companies, since it allows them to obtain detailed feedback from users, and thus know which aspects of their products or services should be improved.

Acknowledgment

Our heartfelt appreciation goes to Professor Dr. Laxmidhar Behera and Sai Surabh Sir with regards to whose feedback across the course of project from the initial proposal up to the conclusion and for the valuable lessons learned along the way including competing within a group and the challenges involved in model tuning.

References

[1] Rui Xia, Chengqing Zong, Shoushan Li, “Ensemble of feature sets and classification algorithms for sentiment classification,” Information Sciences, 181 (6) (15 Mar. 2011), pp. 1138-1152

[2] R. Prabowo, M. Thelwall, “Sentiment analysis: A combined approach,”Journal of Informetrics (2015), pp. 143-157 Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[3] Bing Liu“Sentiment analysis and opinion mining,” Synthesis Lectures on Human Language Technologies, 5 (1) (2012), pp. 1-167

[4] V.S. Jagtap, K. Pawar“Analysis of different approaches to Sentence-Level Sentiment Classification,” International Journal of Scientific Engineering and Technology, 2 (Apr. 2013), pp. 164-170

[5] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding; Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova