Movie Review Sentiment Analysis using Rotten Tomatoes Reviews

Cyrus Tabatabai-Yazdi
UCLA
ctabatab@ucla.edu

Abstract

The ubiquitousness of social media and the Internet has made it easier than ever for people to share opinions and reviews on everything. For businesses, this information is valuable as it allows for them to gauge how people are feeling about their products or ideas. Sentiment analysis is the process of analyzing a piece of text to attach a sentiment to it that represents an opinion that can range from hate to love. This project focuses on understanding the various algorithms and methods used in sentiment analysis by looking at both traditional approaches as well as state of the art approaches. In particular, we have developed a CNN-LSTM neural network architecture to achieve better accuracy in predicting sentiment of the movie reviews. We then corroborate the results by deploying and comparing the traditional machine learning techniques such as Naïve Bayes, SVM, Decision Trees and Random Trees with our hybrid neural network architecture.

1. Introduction

The field of natural language processing is about creating systems that can process and understand language in order to perform certain tasks. One of these tasks is sentiment analysis, the process of determining the emotional tone of a piece of text, which allows us to understand the attitudes and opinions expressed by a person. As can be seen in Figure 1, sentiment analysis has been growing in popularity over the years as businesses see the value in understanding the emotions of people towards their products. Opinions play a very important role in making decisions. People often listen to the opinions and sentiments of other people before making decisions such as buying a product on Amazon or watching a movie. In addition, organizations often conduct surveys or opinion polls to gather feedback on their products and services. With the proliferation of the Internet today, particularly social media, people can now put their reviews and sentiments on anything imaginable. With reviews on nearly everything available online, people and organizations have access to valuable information that can be used for a myriad of purposes from targeted marketing, feedback, and purchasing decisions. Despite the growing amount of interest and large amounts of research, sentiment analysis, like other areas of natural language processing, is still a difficult problem. The human language is complex and teaching computers the various nuances, cultural variations, slangs, sarcasm, and misspellings that exist in languages is a difficult process. Understanding the contextual meaning of words instead of just individual words can help machines give proper overall meaning of a statement and many of the recent state-of-the-art developments in the field involve deep learning techniques.

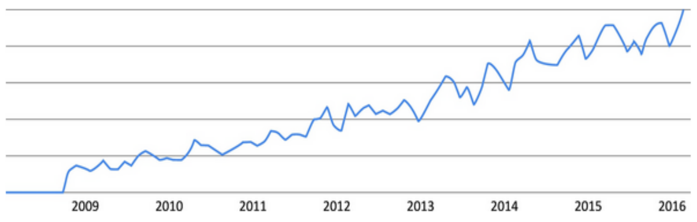

Figure 1: Google Trends data for *sentiment analysis* keyword [1]

2. Related Work

The first paper on using machine learning techniques on sentiment analysis was proposed by Bo Pang et al. [1]. In their research, they considered the problem of classifying a document not by topic but by overall sentiment. They used movie reviews and applied machine learning techniques such as Naïve Bayes, Maximum Entropy classification and Support Vector Machines. Although these techniques outperformed human produced baselines, they still couldn't perform well on sentiment analysis. In a popular paper on Opinion mining, Kushal Dave el al [2] work on classifier that automatically distinguishes between positive and negative review by using information retrieval and feature extraction. Hatzivassiloglou and McKeown [3], in their paper, used a log-linear regression model to identify textual conjunctions such as "fair and legitimate" or "simplistic but well-received" to separate similarly- and oppositely-connoted words. Machine learning techniques have had varying degree of successes. But according to Sida Wang et al [4], a simple but novel SVM variant using NB log-count ratios as feature values consistently performs well across tasks such as topic classification and good and bad sentiment classification. After 2000, due to widespread use of GPUs, deep learning and neural networks have risen in prominence in Natural Language Processing. Bengio et al [5] used a neural network for language modelling that performed better than the state-of-art n-grams model. RNN uses previous state values which is similar and analogous to most of the spoken natural languages. Wang et al. [6] proposed LSTM with Trainable Lookup-Table (LSTM-TLT). The approach bested the state-of-art techniques in Twitter sentiment analysis. The Convolutional Neural Networks(CNN) introduced LeCunn et al. was influential in image and document recognition [7]. DCNN (Dynamic Convolutional Neural Network), a CNN with dynamic k-max pooling layer was proposed by Kalchbrenner et al [8]. DCNN allowed variable length inputs and successfully defeated other models in Twitter sentiment classification.

3. Dataset

The data we are using comes from Rotten Tomatoes. Rotten Tomatoes is a well-known movie review site where professional critics and users can post reviews and the reviews are aggregated to come up with a freshness score from 0-100%. A higher freshness score indicates positive reception. The dataset we are using takes 8529 sentences and parses it into 156060 distinct phrases that each have a sentiment label as shown below:

Label	Sentiment
0	Negative
1	Somewhat Negative
2	Neutral
3	Somewhat Positive
4	Positive

Table 1: Labels associated with Sentiments

Each sentence is divided into an average of 18.30 phrases. The data is not balanced as can be seen below meaning statistical resampling techniques have to be done in order to prevent a biased model.

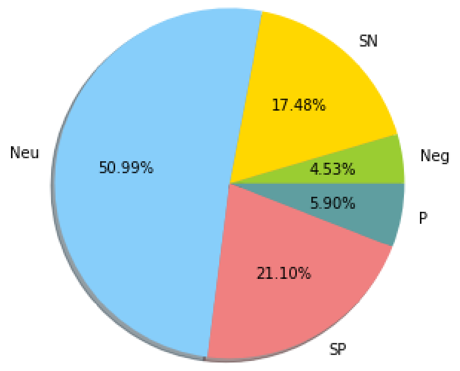

Figure 2: Distribution of Sentiments in Dataset

In our data exploration, we found out the most common words associated with positive, neutral, and negative sentiments. For positive sentiments, words like excellent, fantastic, and beautiful appear. For negative, words like tedious, unfunny, and terrible are the most common.

4. Methods

4.1 Data Preprocessing and Feature Extraction

All machine learning models require input features in the form of numbers, so we cannot just input raw text to a model and expect meaningful results. Thus, we somehow need to extract features that represents the text in our dataset in a meaningful way and without losing any information. Before that, it is important to conduct some data preprocessing as is common in any natural language processing task in order to have clean data to work with. For preprocessing, we converted all the text to lowercase and lemmatized the data. Lemmatizing means to group together variants of the same word meaning words like “thinking” and “think” would be counted as same word in dataset. We decided not to get rid of punctuation as characters like an exclamation mark could be a good marker for sentiment. After the data has been cleaned, the feature extraction process can begin. The traditional approaches to feature extraction from tests is to use Bag-of-Words(BOW) or Term Frequency-Inverse Document Frequency(TF-IDF). The BOW approach looks at all the words that occur in an entire text corpus. In our case, the corpus would be the union of all the phrases in the dataset and the words would be all the unique words that occur. Each phrase in our dataset would be converted to a vector where the length is the number of unique words in our corpus and each entry would be the count for the number of times a particular word occurs in the phrase. For example, consider these three phrases: “*This movie was awesome, and this was cool*”, “*This movie was awesome and cool*”, and “*This movie was bad*”. Using BOW, the matrix representation for the three phrases would be as follows:

	this	movie	was	and	bad	cool	awesome
1	2	1	2	1	0	1	1
2	1	1	1	1	0	1	1
3	1	1	1	0	1	0	0

Table 2: Feature matrix of phrases using BOW

The problem with BOW is that common words like and, is, the, and a may appear a lot and will be large values in the matrix but are not the most useful in sentiment analysis. Thus, the gold standard in text analysis today is to use Term Frequency-Inverse Document Frequency (TF-IDF). It is similar in idea to BOW but weighs down common words that occur in all sentences. The idea is that rarer words like beautiful, fantastic, boring, and terrible are a better way to capture the sentiment of sentences and should be given higher weights. Thus, TF-IDF penalizes the occurrence of common words by giving them low weights while giving higher weights to the rarer words. The summary of TF-IDF is shown below:

$$TFIDF\ score\ for\ term\ i\ in\ document\ j = TF(i,j) * IDF(i)$$

$where$

$IDF = Inverse\ Document\ Frequency$

$TF = Term\ Frequency$

$$TF(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

$$IDF(i) = \log_2\left(\frac{\text{Total documents}}{\text{documents with term i}}\right)$$

and

$t = Term$

$j = Document$

Figure 3: TF-IDF diagram

As explained above, the term frequency denotes the contribution of a word to the document so words relevant to the document should have a higher frequency. The inverse document frequency component penalizes common words that occur in all phrases as those common words may not be relevant.

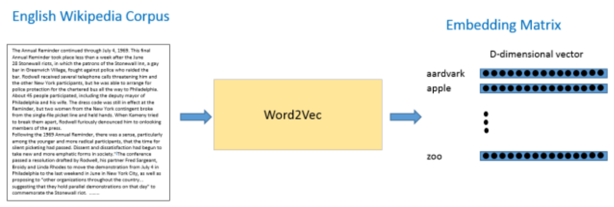

Figure 4: Embedding vector using Word2Vec[2]

TF-IDF does work well, but there is an even better way to represent words as vectors. The idea is to embed each word in an n-dimensional feature space that captures the relationships between words. For example, words like love and adore will be closer to each other in this feature space. There are different algorithms that can be used to learn these word embeddings. The two most popular ones are word2vec and GloVe. For our purposes, we used a pretrained 300-dimensional GloVe word embeddings for 6 billion words trained by Stanford University [6]. The reason is that these embeddings are best when trained on a large corpus such as all of Wikipedia. At a high level, the way these word embeddings work is to consider the context of a word in order to generate vector representations. From the context of the sentences, we can see that words such as love and adore are generally used in sentences with positive connotations and generally precede nouns or noun phrases. This is an indication that both words have something in common and can possibly be synonyms. Context is also very important when considering grammatical structure in sentences. Most sentences will follow traditional paradigms of having verbs follow nouns, adjectives precede nouns, and so on. For this reason, the model is more likely to position nouns in the same general area as other nouns. The model takes in a large dataset of sentences (English Wikipedia for example) and outputs vectors for each unique word in the corpus. The output of a Word2Vec model or GloVe model is called an embedding matrix.

4.2 Naïve Bayes

Naïve Bayes considers the posterior probabilities given the prior probabilities observed in the training set. Naïve Bayes extends Bayes' Theorem to handle this case by assuming that each data point is independent.

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

The above equation determines the probability of class “y” given the features x1, x2 etc. In our project, x is the TF-IDF features that we extracted in the feature extraction and y is one of the sentiment classes. The sentiment class “y” with the maximum probability is assigned the sentiment of the sentence.
The problem that we encountered with Naïve Bayes is that it considers individual words as independent features while in fact most of the words are related to each other in the sentence and can change the meaning of each other depending on where they appear in the sentence.

4.3 Logistic Regression

Logistic regression is a very widely used and historic approach to binary classification. However, it can easily be extended to multi-class classification by using a technique known as one-vs-rest. Very simply, logistic regression outputs the probability of a data point, represented by features, belongs to a specific class. For example, if we are modeling sentiment as being either positive of negative, the first class could be a positive review, and the logistic regression model could be written as the probability of a sentence being positive. The sentence would be represented as a set of features using either TF-IDF or BOW as explained in the previous section.

P (sentiment=positive | sentence)

Binary classification would be useful if we are trying to classify reviews as being either positive or negative. In our case, we are trying to classify a review into 5 classes. Given k different classes, the multiclass logistic regression model would train k different binary classifiers fi(x). We could then predict that an example x belongs to the class i for which the probability that the label applies is highest:

$$\max_i f_i(\boldsymbol{x})$$

To train the model, we minimize the negative log-likelihood through stochastic gradient descent.

4.4 Support Vector Machine

A Support Vector Machine (SVM) is a classifier that defines a hyperplane to categorized examples. In a two-dimensional space, hyperplane is a line that divides the plane into two parts where in each class lies in either side. In our case, we need multiclass classification into labels “Negative”, “Somewhat Negative”, “Neutral”, “Somewhat Positive” and “Positive”. We achieve this by using 5 “One vs All” classifiers and then choose the label with the maximum score.

4.5 Decision Trees

Decision trees build classification or regression models in the form of a tree structure. They break down a dataset into continuously smaller sets while at the same time an associated decision tree is incrementally developed through the ID3 algorithm. The result is a tree with decision nodes and leaf nodes. Each decision node has two or more branches while leaf nodes represent decisions, or in the case of our classification, a sentiment class for each of the input.

4.6 Random Forest

The random forest is an ensemble approach that can also be thought of as a form of nearest neighbor predictor. Ensembles are a divide-and-conquer approach used to improve performance. The main principle behind ensemble methods is that a group of weak learners can come together to form a strong learner. The random forest starts with a standard machine learning technique called a decision tree which, in ensemble terms, corresponds to our weak learner. The random forest is our strong learner. In a decision tree, an input is entered at the top and as it traverses down the tree the data gets put into increasingly smaller sets. When a new input is entered into the system, it is run down all the trees. The result may either be an average or weighted average of all the terminal nodes that are reached, or, in the case of categorical variables, a voting majority

4.7 Deep Learning and Neural Networks

Speech and language recognition have been one of the important applications of neural network and deep learning. The neural networks such as RNNs (Recurrent Neural Networks), LSTMs (Long Short Term Memory) and GRUs (Gated Recurrent Units) are popular for Natural Language Processing and widely used in popular applications such as Alexa and Google Home. Sequence models try to understand the relation between words in a sentence, similar to how humans understand and read a sentence. We try to understand each word in a sentence in context to the previous words or the upcoming words while reading.

4.7.1 LSTM

The problem with traditional RNNs is the vanishing gradient problem which prevents long term dependencies from being captured by the model. For example, take the sentence, “The movie was brilliant, … , and amazing. There could be many words between the two commas so the association between brilliant and amazing may not be captured. This is where LSTMs come into play. An LSTM cell can choose to remember or forget the previous time unit activations, allowing for the creation of a more robust and flexible model that can capture long-term dependencies. The sentence we mentioned may include a lot of filler and non-relevant words between the commas that may have no bearing on the sentiment. sentence had no impact on the question that was asked. However, there is a strong connection between the first part of the sentence and the third part. With a classic RNN, the hidden state vector at the end of the network might have stored more information about the middle part rather than about the first part. Basically, the addition of LSTM units makes it possible to determine the correct and useful information that needs to be stored in the hidden state vector. Each gate will take in x_t and h_{t-1} (not shown in image) as inputs and will perform some computation on them to obtain intermediate states. Each intermediate state gets fed into different pipelines and eventually the information is aggregated to form h_t. For simplicity sake, we won't go into the specific formulations for each gate, but it's worth noting that each of these gates can be thought of as different modules within the LSTM that each have different functions. The input gate determines how much emphasis to put on each of the inputs, the forget gate determines the information that we'll throw away, and the output gate determines the final h_t based on the intermediate states. Looking back at the first example with question “what is the sentiment?”, the model would have to be trained on reviews. The LSTM units would then be able to realize that any sentence without adjectives will likely not have an impact on the answer to the question, and thus the unit will be able to utilize the forget gate to discard the unnecessary information about the filler words, and rather keep the information regarding the sentiments.

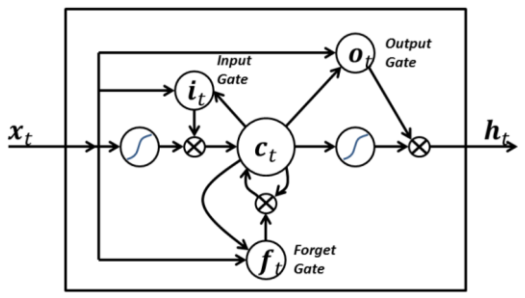

Figure 5: LSTM cell architecture [8]

4.7.2 LSTM and CNN Hybrid

Although CNNs are primarily used for image classification, they are quite efficient in sentence classification by extracting features horizontally from words. This is very useful when words are not in proper order (linguistics allows different parts of speech at different locations of the sentence).

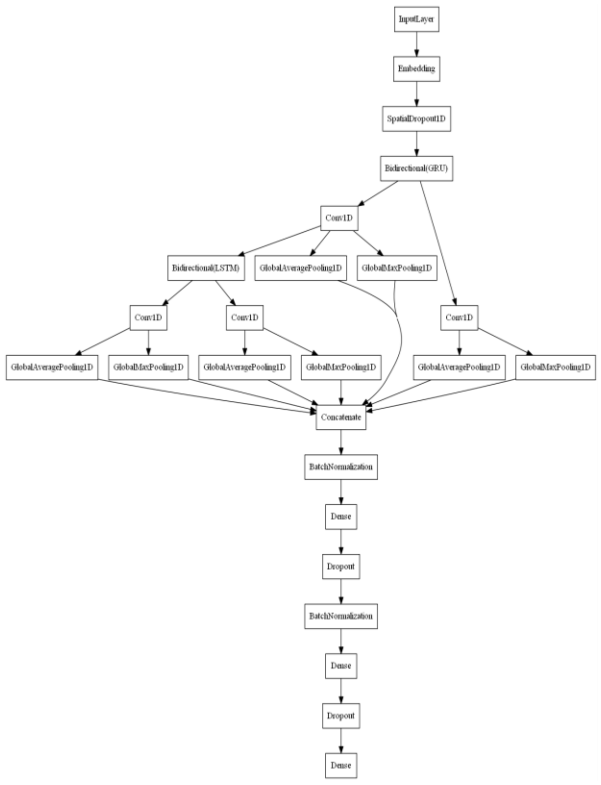

Figure 6: Hybrid Model Architecture

In our case, CNN acts as a feature extractor and LSTM captures the changing sentiment of the review statements [6]. In addition of CNN and LSTM, we used max pooling and average pooling layers to reduce the number of parameters and incorporated a bidirectional LSTM to take into consideration forward and backward dependencies. The final layer is a softmax layer with 5 units that outputs the probability of a sentence/review belonging to each class.

5. Evaluation

To evaluate, we used 10-fold cross validation on the training set to get the average accuracy, precision, and recall. Precision is the number of true positives divided by the sum of the number of true positives and false positives. Recall, meanwhile, is the number of true positives divided by the sum of the true positives and false negatives.

Algorithm	**Accuracy**	**Precision**	**Recall**
Logistic Regression	0.6346	0.6240	0.6356
Support Vector Machine	0.6559	0.6445	0.6559
Naïve Bayes	0.6067	0.6113	0.6067
Decision Tree	0.5818	0.5618	0.5818
Random Forest	0.6838	0.6806	0.6812
Two Layer LSTM	0.7354	0.7304	0.7354
LSTM + CNN	0.8679	0.8542	0.8697

Table 3: Performance metrics of models

As you can see, the traditional machine learning algorithms used for classification do not perform poorly. Considering the fact this is a 5-class classification problem, the results are decent. However, the deep learning models, both the 2-layer LSTM model and the hybrid LSTM + CNN model perform better with the hybrid model performing incredibly well. This is not a surprise as ensemble approaches where different algorithms are applied to a problem seems to be the norm in many machine learning problems.

We display the confusion matrices for the best performing non-deep learning model and the hybrid LSTM/CNN model in Table 3 and Table 4 respectively. The random forest model classified labels 1-3 mostly correct. The issue was with labels 0 and 4(negative and positive). For those labels, most were incorrectly classified as somewhat negative and somewhat positive respectively. This is not as bad as a mistake since the model correctly predicted it was a negative or positive type review. However, the hybrid model was better able to discern between these two close labels.

	0	1	2	3	4
0	764	573	52	0	0
1	351	3983	1074	27	2
2	19	1119	13712	987	17
3	5	57	1030	5135	462
4	0	0	26	505	1381

Table 4: Hybrid Model Confusion Matrix

	0	1	2	3	4
0	609	684	95	8	1
1	485	3246	1560	145	1
2	112	1621	12665	1417	38
3	2	171	1985	4021	510
4	0	14	76	945	801

Table 5: Random Forest Confusion Matrix

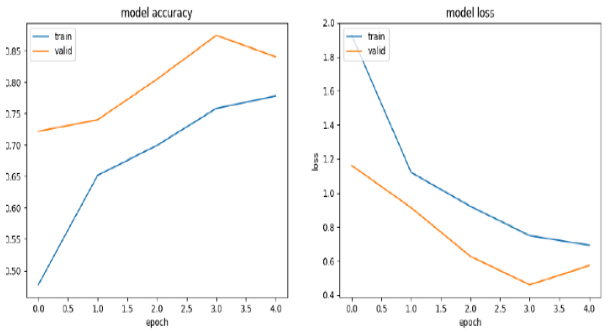

Figure 7: Hybrid Model Accuracy and Loss

In Figure 7, we show the accuracy and loss when training the hybrid model. We also generated Receiver Operating Characteristic Curves for further insight into how well the various algorithms perform. The Receiver Operating Characteristic (ROC) curves for each class for the Random Forest and Hybrid Deep Neural Network models are shown below. A higher area under the curve is desirable and the hybrid deep learning model outperforms the random forest in all instances.

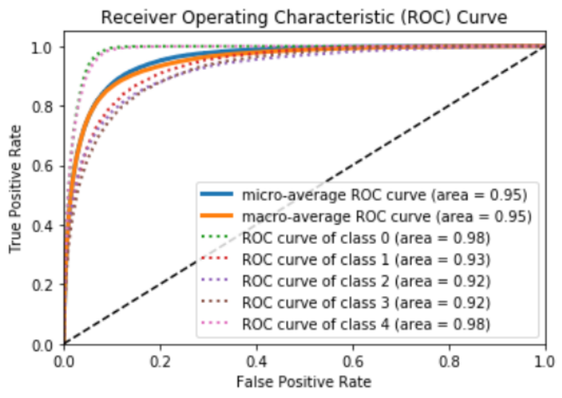

Figure 8: Hybrid Model ROC Curve

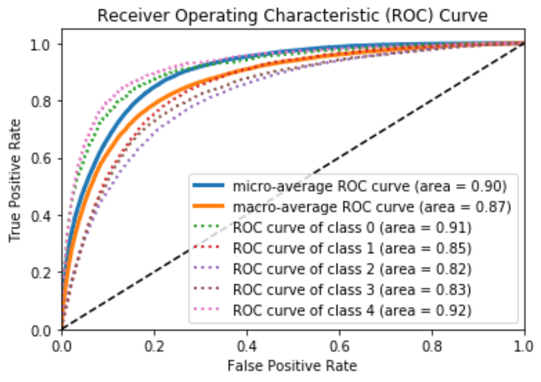

Figure 9: Random Forest ROC Curve

6. Future Work

We have achieved 86% accuracy with our hybrid CNN-LSTM neural networks. We plan to improve upon the current architecture by looking at other types of architectures such as variational autoencoders. Another approach is to do a combination of Naïve Bayes and neural networks. We can also linguistics domain knowledge to better understand a field. For example, medicine vocabulary has different meanings for the same meaning. Like “positive” in common language implies a positive sentiment but in medical terms, it could mean a negative sentiment upon disease discovery. Our project only considered text, but in the future, we could look at facial expressions along with the spoken words to get track the idiosyncrasies of the spoken language such as sarcasm, humor etc. This would mean incorporating computer vision techniques to gauge emotion from the image of a face.

7. Conclusion

The applications of sentiment analysis are broad and powerful. The ability to extract insights from social and web data is a practice that is being widely adopted by organizations across the world. For example, shifts in sentiment on social media have been shown to correlate with shifts in the stock market [10]. The Obama administration used sentiment analysis to gauge public opinion to policy announcements and campaign messages ahead of 2012 presidential election [9]. Being able to quickly see the sentiment behind everything from forum posts to news articles means being better able to strategize and plan for the future. It can also be an essential part of market research and understanding customer opinions. Not only can it be seen what people think of your own products or services, but also can allow insight into what people think about competitors. The overall customer experience of your users can be revealed quickly with sentiment analysis, but it can get far more granular too. Most of the recent advances in sentiment analysis rely on deep learning techniques, especially deep natural language processing and future advances will likely follow in this trend.

8. References

[1] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? sentiment classification using machine learning techniques." *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10.* Association for Computational Linguistics, 2002.

[2] Dave, Kushal, Steve Lawrence, and David M. Pennock. "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews." *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003.

[3] Hatzivassiloglou, Vasileios, and Kathleen R. McKeown. "Predicting the semantic orientation of adjectives." *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*. Association for Computational Linguistics, 1997.

[4] Wang, Sida, and Christopher D. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification." *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*: Short Papers-Volume 2. Association for Computational Linguistics, 2012.

[5] Bengio, Yoshua, et al. "A neural probabilistic language model." *Journal of machine learning research* 3.Feb (2003): 1137-1155.

[6] Wang, Xin, et al. "Predicting polarities of tweets by composing word embeddings with long short-term memory." *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Vol. 1. 2015.

[7] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

[8] LSTM Wikipedia page, https://en.wikipedia.org/wiki/Long_short-term_memory

[9] Pak, Alexander, and Patrick Paroubek. "Twitter as a corpus for sentiment analysis and opinion mining." *LREC*. Vol. 10. No. 2010. 2010.

[10] Kouloumpis, Efthymios, Theresa Wilson, and Johanna D. Moore. "Twitter sentiment analysis: The good the bad and the omg!." *Icwsm* 11.538-541 (2011): 164.

[11] Nakov, Preslav, et al. "SemEval-2016 task 4: Sentiment analysis in Twitter." *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*. 2016.

[12] Gilbert, CJ Hutto Eric. "Vader: A parsimonious rule-based model for sentiment analysis of social media text." *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*. 2014.