Predicting Stock Movement through Executive Tweets

Michael Jermann
Department of Computer Science
Stanford University
Palo Alto, CA 94305
mjermann@stanford.edu

Abstract

This paper details an investigation into the viability of predicting stock prices directly from the tweets of company executives. I begin by describing the data collection process and the natural language processing framework. The paper then documents the construction, tuning, and evaluation of the various neural network models and their performance. In the end, a very minor classification accuracy gain is achieved, but I believe the results serve as a promising indication that with additional data collection increased accuracy could be achieved.

1 Introduction

In an age where company executives can achieve a degree of fame comparable to actors, athletes, and musicians, their social media activity is often scrutinized. An announcement, opinion, or remark can be enough launch a news story or directly influence the market.

In this paper I use publicly available data from one such social media platform, Twitter, and perform natural language processing (NLP) on it in order to predict positive and negative fluctuations in stock price based on the text. Many existing efforts have shown that this is possible on a macro scale by looking at ecosystem-level trends; however, I take the approach of identifying a small subset of influential Twitter accounts and analyzing their tweets with a variety of NLP techniques. Both sentence-level and word-level language features are used, and the NLP pipeline is built to consider Twitter-specific syntax (such as hashtags, emoticons, and mentions). This results in a neural-network model that achieves a minor gain over baseline (+0.1%) but nonetheless serves as a proof-of-concept that this can be a viable trading strategy with an increased executive list and additional training examples.

2 Background/Related Work

There has been significant interest both in tailoring natural language processing tasks to Tweets and using Twitter data to predict stock market fluctuations, much of which this paper builds upon. Much of the research regarding Twitter NLP has dealt with sentiment classification, which is a similar task to the one undertaken here. Researchers have developed methods for training sentiment-aware word embeddings [1]-[4] which are useful vector space representations of words used in a Twitter context. By understanding the Twitter syntax and leveraging the huge amount of data available via the Twitter API, these researchers have developed several approaches for encoding the emotional subtext of Tweets into the word embeddings without extensive manual labeling. Additionally, the identification and engineering of high-performing Twitter-specific features such as sentiment association lexicons [5], twitter-specific part-of-speech identification [6]-[7], and entity and event extraction [8]-[10] have increased the amount of available information regarding tweet content and semantic information that can be harnessed for prediction. Using these types of features and word

embeddings, convolutional neural networks have been demonstrated to be very effective at parsing this type of text for sentiment prediction [11].

Research into the relationship between Twitter and the stock market has identified a clear link between the two as well. In a ground-breaking analysis, predicted "calmness" based on large-scale Twitter activity was demonstrated to be correlated with the stock market [12], which subsequent analysis corroborated [13]. Performing NLP on company financial reports has also been shown to significantly improve stock price predictions [14], indicating that a similar type of model may be effective if adapted to Twitter.

3 Approach

While much of the existing research has taken a macro approach to predicting the stock market with Twitter (i.e. aggregating data from the ecosystem), I took a micro approach by identifying a set of "high signal" Twitter users (verified executives at publicly traded corporations) and attempted to process the text content of their tweets to directly predict stock movements (up or down). While it is obviously impractical to expect to achieve a high level of accuracy on this task (given that the majority of stock movements are not caused by tweets, and vice versa), the goal of this paper is to serve as a proof-of-concept that the Twitter activity of a highly-curated list of business executives can be used as a signal for stock market prediction.

I leveraged the "best practices" established in Twitter sentiment research to build a custom NLP processing pipeline that includes tokenization, word vector representations and sentiment lexical scores. The classification models developed are primarily neural networks in which the words comprising the tweet text (parsed with a tokenizer built to handle Twitter syntax) are represented by vector embeddings. Sentence-level features will also be considered (e.g. "containsEmoticon" or "hashExclamation"), with the goal of using both as inputs to a multi-layered neural network which achieves a high level of classification accuracy over the baseline (i.e. a Naive Bayes bag-of-words classification model).

4 Experiments

This section describes the end-to-end process of obtaining and processing the data, performing exploratory data analysis to formalize the classification algorithm, and iterating upon versions of the neural network classification models.

4.1 Data Collection

The data collection process used to generate this dataset is detailed as follows.

4.1.1 Twitter Accounts

I began with a list of all NASDAQ and NYSE companies (6,405 in total, obtained via Quandl[1]), containing both the stock symbol and the company name. I compiled an initial list of relevant executives through a combination of querying the ISS Directors database[2] and manual inspection. I then queried the Twitter API[3] to search for each name. For each name, I iterated through the search results (ordered by descending number of followers) and returned the ID of the first user who was verified and contained the name of the company in their description. I then manually inspected the list (removing non-English users, false positives, and low-activity members). The list of Twitter accounts followed by these accounts was then crawled, with all accounts containing the company name being returned, followed by additional manual inspection. After several iterations, I obtained a total of 106 user accounts representing 44 companies. I then queried these accounts for public tweets (including retweets and replies), resulting in 84,859 tweets. Metadata regarding these tweets was also recorded (such as hashtags, mentions, and number of followers).

[1] https://blog.quandl.com/useful-lists

[2] http://www.whartonwrds.com/datasets/iss/

[3] https://python-twitter.readthedocs.io/en/latest/

Table 1: Dataset

Set	Number of Samples	Percentage POS
Train	26,171	51.6%
Development	3,189	51.8%
Test	3,182	51.9%

4.1.2 Stock Data

Half-hourly stock data for each company was purchased via Kibot[4]. This was used to calculate the half-hour change in stock price following the tweet creation. This timeframe was used in order to isolate the effect of the Tweet on the stock market. For tweets made during off-hours, the next available stock price was used. For each company, the text of all tweets created within a half-hour (regardless of author) were concatenated and features such as "numberOfTweeters" and "numberOfTweets" were calculated. This resulted in a dataset of 32,542 training examples.

4.2 Formulation

As seen in Figure 1, the vast majority of the stock fluctuations were very small. Therefore, directly identifying the major shifts would be difficult - with less than 5% of the dataset seeing an increase of more than 1%, more training examples would be required to for a network to learn to effectively identify these.

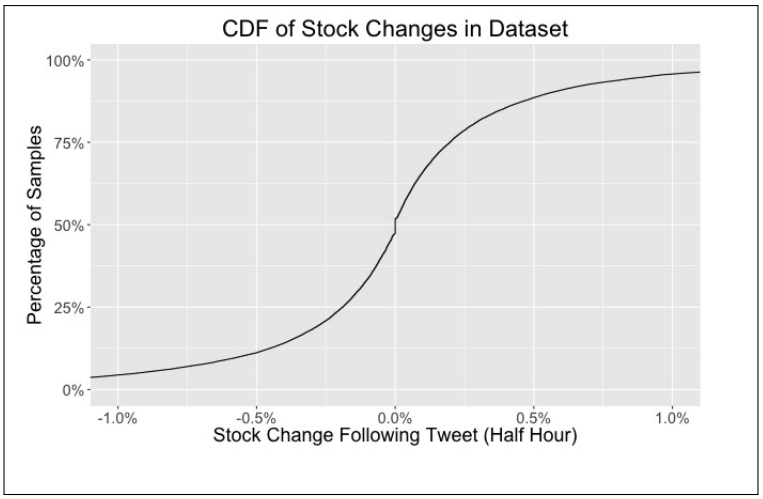

Figure 1: Cumulative distribution function for stock price changes.

Therefore, I elected to perform binary classification with a threshold of zero given that identifying any increase over baseline accuracy would demonstrate proof of concept. Those with an increase greater than or equal to 0.0 were labeled as positive whereas those with an increase less than 0.0 were labeled as negative.

The dataset was divided into train, development, and test sets constituting roughly 70%/15%/15% of the dataset. The training set was used to train the model parameters, the development set was used to optimize the hyperparameters, and the test set was used **only** on the model that performs best on the development set in order to gauge if the model is generalizable.

The metric used for evaluation was be classification accuracy. This basic metric is more appropriate than something such as F1 score given that I am predicting the labels of classes with roughly

[4]http://www.kibot.com/

Table 2: Tokenization Methods

Method	Sample
NLTK	”she”, ”’s”, ”back”, ”,” , ”for”, ”this”, ”week”, ”only”, ”!”, ”#”, ”blessed”, ”.”, ”:”, ”)”
Glove	”she”, ”’s”, ”back”, ”,”, ”for”, ”this”, ”week”, ”only”, ”!”, ”<hashtag>”, ”blessed”, ”.”, ”<smile>”
Twokenize	”she’s”, ”back”, ”,”, ”for”, ”this”, ”week”, ”only”, ”!”, ”#blessed”, ”.”, ”:)”

Table 3: Accuracy of Top Naive Bayes Model

Train	Dev	Test
51.6%	52.0%	52.1%

equivalent sizes. As seen in in Table 1, any accuracy over 51.9% on the test set is an increase over predicting all positives.

4.3 Word

The word-based models consisted primarily of tokenizing the string of the tweet into modular components to be represented by a vector, and then performing operations on those vectors to predict the label of the training example. This section details the experimentation regarding these models.

4.3.1 Baseline

For a baseline, I trained a Naive Bayes model with a bag-of-words approach (i.e. representing each training example by a set of the words it contains). To parse the text into words, I used several tokenization methods, two of which could recognize Twitter-specific punctuation and syntax (which is vital to maximizing the number of words that can be accurately represent with vectors). The three tokenizers used were from Natural Language Toolkit python platform[5], an adaptation of the Glove tokenizer script[6] (with several modifications from Christopher Potts’ Twitter tokenizer[7]) [15], and Twokenize[8] [16] which was published by the Carnegie Mellon NLP group. Sample tokenization of the string ”She’s back, for this week only! #Blessed. :)” are contained in Table 2.

A randomized grid search in which the tokenizer was varied (along with many of the hyperparameters) led to a Naive Bayes classifier with the performance listed in Table 2. The best model used the NLTK tokenizer, did not ignore ”stop words” (or very common words that typically have low predictive value), and represented a slight improvement over guessing ”positive” for every label. It achieved 52.1% accuracy on the test set, or a modest 0.2% increase. While this change is very small, it is of the expected magnitude and provides evidence that there is at least mild signal here. This also served to temper expectations as to the performance of more complicated models on this dataset.

4.3.2 Embeddings

In order to ascertain if further gains could be achieved by using a vector-space representation of words, my next step was to map words to a vector space representation. I considered two types of pretrained embeddings that were trained on Twitter data: Glove[9] [17] and Sentiment-Specific Word Embeddings (SSWE)[10]. These embeddings differ based on how they treat punctuation and twitter syntax but align with the Glove and CMU tokenizers, respectively, so each was only used with the

[5]http://www.nltk.org/api/nltk.tokenize.html

[6]https://nlp.stanford.edu/projects/glove/preprocess-twitter.rb

[7]http://sentiment.christopherpotts.net/code-data/happyfuntokenizing.py

[8]https://github.com/brendano/ark-tweet-nlp/

[9]http://nlp.stanford.edu/data/glove.twitter.27B.zip

[10]http://ir.hit.edu.cn/ dytang/paper/sswe/14ACL.pdf

appropriate tokenizer. Both represent tokens with 50-dimensional vectors. Therefore, for the initial model the input to the neural network was merely the average of the word vectors of each token found within the text (i.e. 50 inputs) fed into a softmax layer to predict positive or negative. For both embeddings the training set saw a dramatic increase in accuracy on the training set (but not the dev set), indicating overfitting. This was a positive sign in that it indicated that the input dataset was sufficiently complex as to be able to represent the training data. On the next iteration I added a L2 regularization term into the cost function, which successfully reduced overfitting (as seen in Figures 2 and 3). This had a more beneficial effect on the model using the SSWE embeddings, which was able to achieve an accuracy on the test set of 52.2%, or a 0.3% increase over guessing all "positive" (and a 0.1% increase over the established baseline), as seen in Table 4.

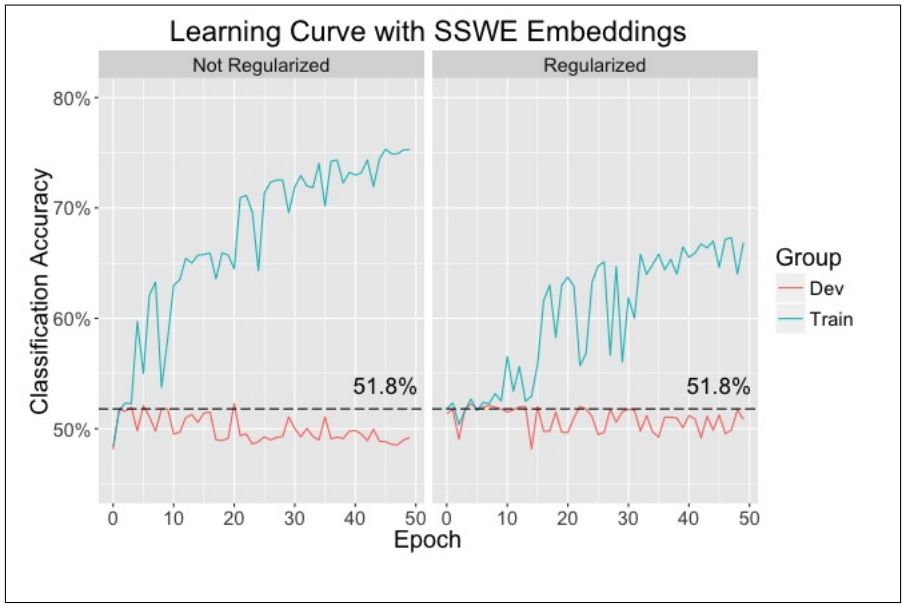

Figure 2: SSWE embedding learning curve with and without regularization.

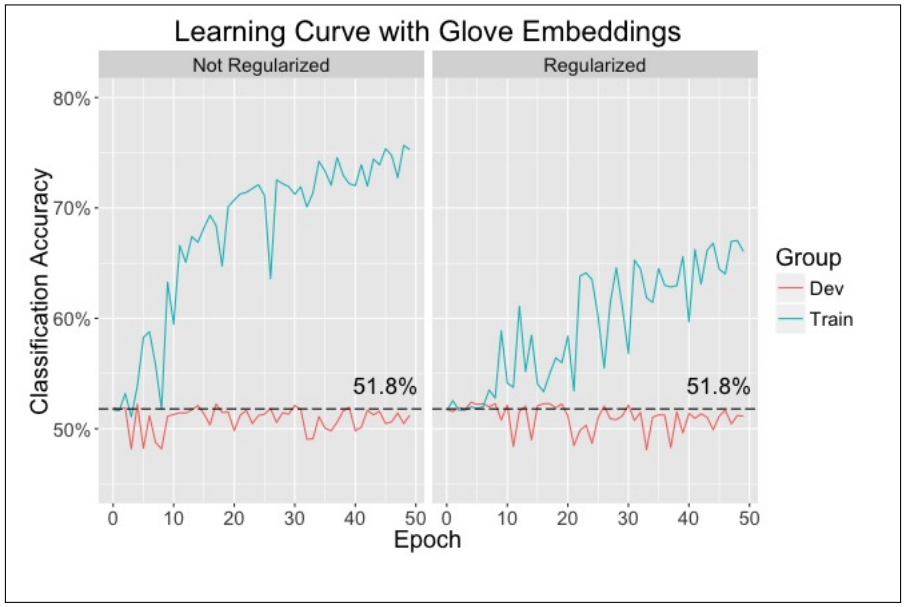

Figure 3: Glove embedding learning curve with and without regularization.

Table 4: Accuracy of Top Embeddings Model

Regularization	Embedding	Train	Dev	Test
No	Glove	53.9%	52.2%	51.6%
No	SSWE	55.0%	52.1%	51.3%
Yes	Glove	51.8%	52.2%	51.9%
Yes	SSWE	52.7%	52.3%	52.2%

I also attempted including both trainable and untrainable embeddings along with combinations of these, but significant performance improvements were not observed.

4.3.3 Convolutional Neural Network

In order to consider each word's position within the text and its proximity to other words, I next attempted a model in which the embedding vector values were not merely averaged. Recent research [11] has found that convolutional neural networks are extremely effective at tasks such as sentiment parsing. Given that sentiment prediction is similar to this task, I implemented a convolutional neural network to see if it resulted in any performance increase. Much of the code for this portion was adapted from the TFLearn sample code[11]. This network was implemented by tokenizing the input string and padding each list of inputs to the same length (that of the largest input, enabling me to process the input as a batch). This input was then fed to 3, 4, and 5-input convolutional units, merged, and combined via a max pool unit. This output then goes through a hidden layer before being input to a softmax layer for prediction. The logic behind this is that any especially "important" sequences of 3-5 tokens will drive the output. This is very prone to overfitting, so the model was trained with dropout (20%) and L2 regularization. Unfortunately these were insufficient in combating the dramatic overfitting that occurred in these models as seen in Figure 4.

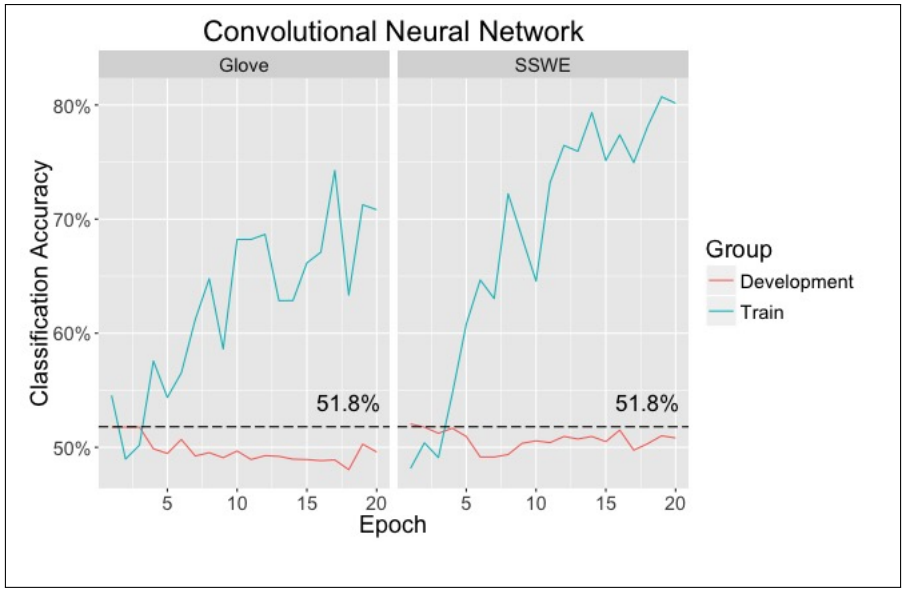

Figure 4: Learning curve for convolutional neural network.

[11] https://github.com/tflearn/tflearn

4.4 Sentence

An orthogonal set of features by which to evaluate the contents of these tweets are sentence-level features. Sentiment analysis has successfully been performed on tweets using sentence-level features and a classification algorithm [5] so it was logical to attempt this approach concurrently.

4.4.1 Baseline

For the baseline, I trained a logistic regression (i.e. a one-layer neural network with a softmax activation function) to predict the two classes. The features used were easily obtainable from the training data and selected through a combination of intuition and exploratory analysis. The logistic regression achieved a poor fit, not even able to fit well to the training data.

Given that the model was unable to fit the training data, it seemed that the the features selected were insufficient to model the underlying data. Therefore, I added some features regarding "sentiment" scores for unigrams and bigrams [5][12] to augment the existing feature set. This provided minor improvement (52.0%) on test set, but underperformed the Naive Bayes baseline.

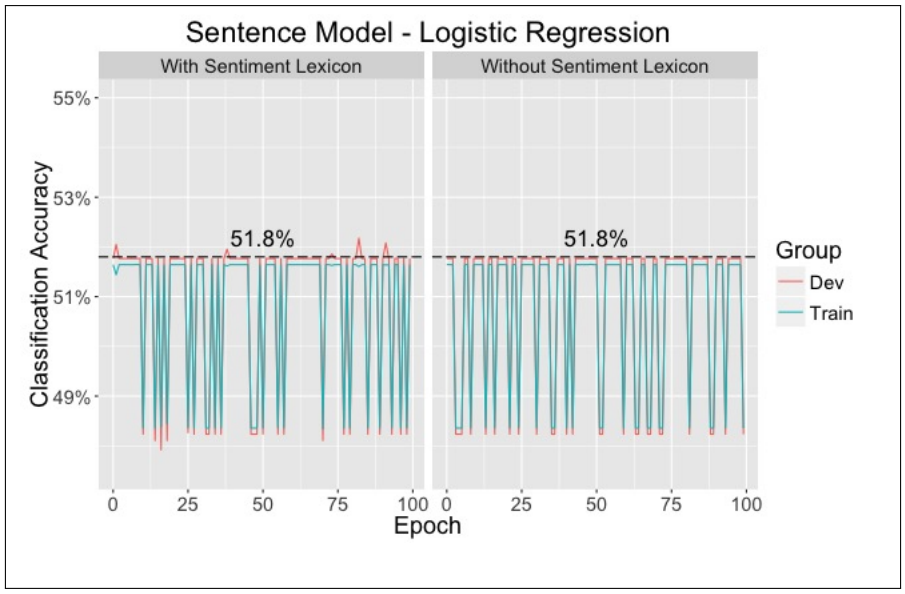

Figure 5: Sentence level model performance curve.

4.4.2 Hidden Layer

Since we saw minor improvement after adding the advanced features, I attempted to add a hidden layer with a moderate number of nodes (roughly two times that of the number of sentence level features) and a ReLU activation function. The idea is that cross-terms between the input features may expose interdependencies that create more complexity. With the extra layer, I added a dropout node (with various keep probabilities) to help guard against overfitting. Regardless, there was little improvement and cost actually decreased more slowly.

4.5 Combined

Given that the sentence level features did achieve some success, I consolidated the best performing word and sentence classifiers into a two layer neural network. Basically, the average of the embeddings is concatenated with the sentence level features and fed to a softmax classifier. While the development set reached accuracy levels comparable to that of the best word and sentence model

[12] http://saifmohammad.com/WebPages/Abstracts/NRC-SentimentAnalysis.htm#download

iterations, the best performance on the test set was 52.0% - the same as the logistic sentence-level baseline and worse than the Naive Bayes baseline.

5 Conclusion

Despite the best model only achieving a classification accuracy 0.3% higher than predicting every tweet would result in an increase, I think that these results indicate there is signal here that could be better harnessed by increasing the dataset to encompass more executives. This dataset only had around 100 Twitter users representing 40 companies; a curated list would have enabled me to discard many of the personal and unrelated tweets used here (which caused unnecessary noise) and potentially improve performance. The most obvious next step is to expand this list and better filter out non-business tweets.

This project also demonstrated the utility of custom-trained word embeddings. Using word vector representations even in an unsophisticated manner (merely averaging them) was shown to be more effective than implementing advanced techniques such as convolutional neural networks and multi-layered neural networks. Another interesting follow-up from this research would be to train custom embeddings tailored to this prediction task.

References

[1] Tang, Duyu, et al. "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification." ACL (1). 2014.

[2] Tang, Duyu, et al. "Coooolll: A deep learning system for twitter sentiment classification." Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). 2014.

[3] Severyn, Aliaksei, and Alessandro Moschitti. "Twitter sentiment analysis with deep convolutional neural networks." Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2015.

[4] Maas, Andrew L., et al. "Learning word vectors for sentiment analysis." Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011.

[5] Mohammad, Saif M., Svetlana Kiritchenko, and Xiaodan Zhu. "NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets." arXiv preprint arXiv:1308.6242 (2013).

[6] Agarwal, Apoorv, et al. "Sentiment analysis of twitter data." Proceedings of the workshop on languages in social media. Association for Computational Linguistics, 2011.

[7] Owoputi, Olutobi, et al. "Improved part-of-speech tagging for online conversational text with word clusters." Association for Computational Linguistics, 2013.

[8] Saif, Hassan, Yulan He, and Harith Alani. "Semantic sentiment analysis of twitter." International Semantic Web Conference. Springer Berlin Heidelberg, 2012.

[9] Ritter, Alan, Sam Clark, and Oren Etzioni. "Named entity recognition in tweets: an experimental study." Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2011.

[10] Ritter, Alan, Oren Etzioni, and Sam Clark. "Open domain event extraction from twitter." Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012.

[11] Dos Santos, Ccero Nogueira, and Maira Gatti. "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts." COLING. 2014.

[12] Bollen, Johan, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." Journal of computational science 2.1 (2011): 1-8.

[13] Chen, Ray, and Marius Lazer. "Sentiment analysis of twitter feeds for the prediction of stock market movement." stanford. edu. Retrieved January 25 (2013): 2013.

[14] Lee, Heeyoung, et al. "On the Importance of Text Analysis for Stock Price Prediction." LREC. 2014.

[15] Potts, Christopher. "Sentiment Symposium Tutorial: Tokenizing." Sentiment Symposium Tutorial: Tokenizing. N.p., 8 Nov. 2011. Web. 20 Mar. 2017.

[16] "Tweet NLP." Twitter Natural Language Processing – Noah's ARK. N.p., n.d. Web. 20 Mar. 2017.

[17] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." EMNLP. Vol. 14. 2014.