Stan Thornhill
Master Project Report
Advisor: Jay Pujara

Stock Market Forecast using Semantic Orientation of Tweets

Introduction

The prediction of stock prices is considered a very difficult task. Prediction models are used to determine investment entry and exit points. These tools are invaluable to stock traders and analysts, but traditional methods do not incorporate the aggregate sentiment of the microblog community.

There are statistical models or predictive computational models to predict the stock market. Simple statistical models, such as moving averages, use mathematical formulas to smooth the price movements which reveal a stock's price trend. Other statistical models, such as ARMA, which one of many models for time series modeling, combines an autoregressive model with moving average to make its prediction [6]. Computational models, such as SVMs, have been used to make predictions on price movement directions where features such as trading volume, past prices, and macroeconomic variables are used [1].

The efficient-markets hypothesis states that a market is efficient when all information is completely available to be incorporated into market prices. It has been found, however, that people's interpretation of the available information has an effect on the market. Recently, the adaptive-markets hypothesis has been used to describe stock market efficiency. It states that there exists a nonlinear dependence relationship between the efficient-markets hypothesis and behavioral finance, where market returns are a consequence of human behaviors. The hypothesis states that market returns go through periods of independence and dependence [1].

In the paper, "Does the Stock Market Overreact?" [12], DeBond and Thaler found that investors could be subject to waves of optimism and pessimism, which would cause prices to deviate from their reported earnings. More recently, in the paper "More Than Words: Quantifying Language to Measure Firms' Fundamentals" [13], the researchers found a link between textual sentiment and the economy. They found that negative words in the financial press forecast low firm earnings.

A large amount of information about a market and people's' opinions about the market is in the form of unstructured data such as financial reports from banks, company earnings reports, newspapers, and social media [1].

Researchers have found that social media has allowed them to measure, in real time, the attention, views, and feelings of a large population. For example, Twitter was used to mine public sentiment of the 2014 Ferguson Riots. Twitter has also been used to make predictions. For example, in the paper, "Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment" [14], were able to predict the outcome of the German national parliament election of 2009. They found that the number of messages mentioning a party reflected the election results.

In regards to finance, social media provides an up-to-date insight of opinions and the sentiment of consumers and investors. Microbloggers express sentiment of a company price movements. Thomas Renult, a finance doctoral candidate and an author at The Conversation, states that this research is still in its infancy which could lead to new discoveries in the field of economics [15]. The goal of much of this research is to find possible relationships between the unstructured text data, people's perceptions, and the economy.

Problem Statement

The problem addressed in this paper is how can the sentiment of tweets be accurately captured for stock market prediction. By improving our accuracy of the “feeling” of tweets, we can see the effect it has on our ability to predict the stock market. The aim of this paper is to build a model that is more robust than current approaches.

The Semantic Orientation (SO) algorithm has been used in the past for sentiment analysis. Given words that are polar opposites to each other, SO can be used to determine where text fall in that spectrum. Current approaches to the SO algorithm use PMI to determine the association between words. It calculates probabilities based on word occurrences that assumes independence between pairs of words. Instead of using PMI, this paper proposes a word embeddings model to express these associations. By using word embeddings to calculate the semantic orientation of polar opposite words, an aggregate sentiment of the twitter population can be obtained. That semantic orientation is incorporated into a prediction model and compared to the PMI-based approach.

In the following sections I will provide background information about semantic orientation and word2vec based word embeddings, an initial experiment, my proposed modification to word embeddings model, additional experiments run, and analysis.

Background

Semantic Orientation algorithm is used to return a score for a word that says where that word lies on a particular spectrum [2]. In online-text-mining for market prediction it has been used to determine bearish/bullish sentiment of the population by averaging the score of nouns, adjectives, adverbs, verbs used for that day [16]. The SO algorithm uses pointwise mutual information to measure the association between two words. It is expressed by this equation:

$$SO(w) = PMI(w, a_1) - PMI(w, a_2)$$

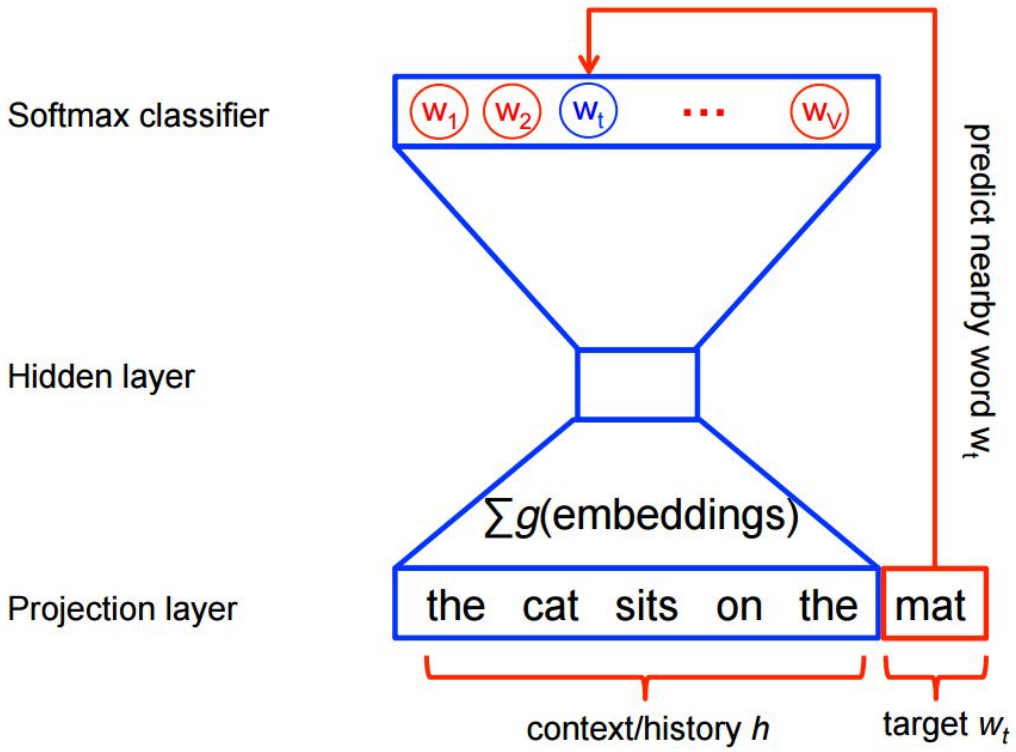
The variables a_1 and a_2 are anchor words that are antonyms of each other. PMI is defined :

$$PMI(w1, w2) = log\frac{p(w1, w2)}{p(w1)p(w2)} = log\frac{p(w1|w2)}{p(w1)}$$

PMI expresses the dependence between words by counting co-occurrences. For example, "Costa" and "Rica" have a high PMI. The SO where $a1 = \text{"bullish"}$ and $a2 = \text{"bearish"}$ would give a positive or negative value where w in the spectrum of bullish and bearish.

The next concept used in this paper is word2vec based word embeddings. A word embeddings model is a model that maps a vocabulary into a low-dimensional vector representations of continuous space. This is different than representing words as unique discrete ids that is done with the n-gram model. In this vector space semantically similar words have embeddings near one another.

Word2Vec, a recently developed word embeddings model [11], has become a popular model to generate word embeddings. It follows the Distributional Hypothesis, which states that words that appear in the same contexts share semantic meaning. Word2Vec a feed-forward neural network to predict a target word given as input words in context around the target word. Figure 1, taken from the TensorFlow word2vec tutorial [7], shows a visualization of this design.



Given the sentence "The cat sits on the mat.", "the cat sits on the" is the input and "mat" is target word. The neural network is trained on the input sentences, target vocabulary, and a context window size which says how many words to consider to the left and right of the word. First, each word is mapped to an embedding. The embeddings are initially randomly assigned. The embedding layer has an arbitrary size, and the output layer has the size 1 + the number of negative samples. For the output layer n number of words is sampled from a modified unigram distribution. For more detailed explanation of this design see Mikolov et al [11].

The output layer runs through a softmax layer that transforms the activations to probabilities. Given a context word as input, the objective is to maximize the output probability of the target word, and minimize the probabilities of the noise words. This could be considered as multinomial logistic classification.

Once the model is trained, a word is fed forward where the activation vector of the hidden layer is the word embedding of that word. The model of representing the context words as input and a target word as output, as shown in Figure 1, is called the Continuous Bag-of-Words model (CBOW). An alternative approach is to represent the target word as the input and have each source context word as separate outputs. This is flavor of word2vec is called the skip-gram model. The objective is same in both models. Among word2vec's contributions, it has been shown to improve sentiment analysis [10].

Initial Experiment

In an initial experiment a week's worth of twitter data was collected to see how well it could be used to the SO algorithm. For each day, the SO was calculated for all words of that day with respect to the words being related to "bullish" and "bearish". A pre-trained word2vec data file was used that was originally trained from Google News dataset (about 100 billion words).

The results showed that the aggregate sum of "bullish" and the aggregate sum of "bearish" was very similar for each day. As a result, the aggregate SO for each day was very close to 0. Upon further inspection, it was discovered that "bullish" and "bearish" have a cosine similarity of .88. It was originally hypothesized that since they are antonyms, they would have a negative similarity value.

Since the similarity between the polar opposite words that define the spectrum is close, it can be assumed the distance all words are to “bullish” and “bearish” will also be relatively close to being equal. This close equality can mean that perturbations in the vector can change which side of the spectrum to word falls in. As a result, this paper implements a modification to word2vec in order to make antonymous words far apart from each other.

Word2vec loss modification

Masataka Ono, et al [9] found that word embeddings that follow the distributional hypothesis often fail to recognize antonyms since antonymous words, occur in similar contexts, e.g. strong and weak. In their research the goal was to find antonym/synonyms not originally expressed in WordNet. They proposed using the same neural network model as word2vec, but change the objective. The original objective function can be expressed as follows:

$$\sum_{w \in V} \sum_{c \in V} \#(w,c) log(\sigma(sim(w,c))) + k\#(w) P_o(c) log(\sigma(-sim(w,c)))$$

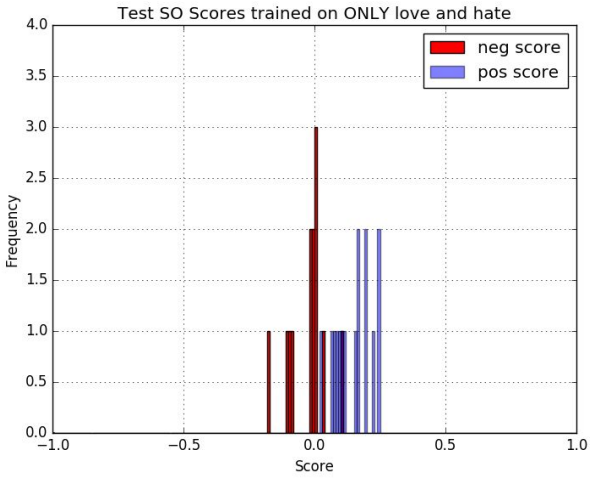
The first term in the summation represents the amount similarity between a target word w and the words in context around that target word c. It is number of co-occurrences of w and c times the log of the sigmoid of the cosine similarity between w and c. The second term in the summation represents the negative sampling of w. k is the number of negative samples in the output layer, $\#(w)$ is the number of occurrences of w as a target word, and $P_o(c)$ is an alternated unigram distribution that considers the proportion of occurrences of a word in the corpus.

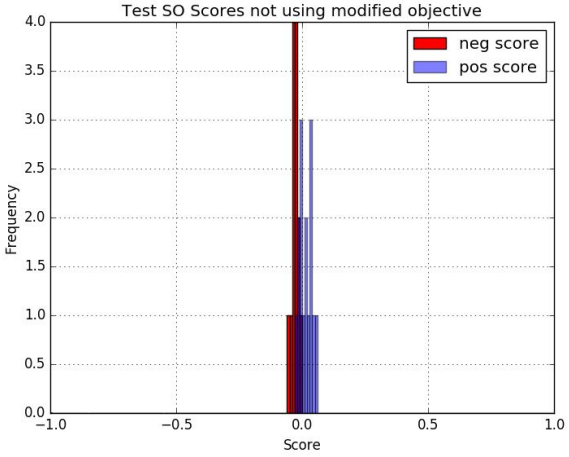
The authors proposed adding two additional terms to the objective:

$$\sum_{w \in V} \sum_{s \in S_w} log(\sigma sim(w, s)) + \sum_{w \in V} \sum_{a \in A_w} log(\sigma(-sim(w, a)))$$

Here S_w represents synonyms of a word in WordNet and A_w represents antonyms of w also found in Wordnet. Adding these terms to the original loss favor similarity between synonyms and dissimilarity between antonyms. Similar to their research, the proposed modification is to use a lexicon of two sets of words where one set is decided to be polar opposites of another set. In this case, a word is considered a synonym to all words in its set and an antonym to all words in the other set. The objective is maximized when all words of a set have high similarity to each other and low similarity to all words in the other set. By having such a word embeddings model, the goal would be to more accurately find out where a word, that's not in the polarized lexicon, lies relative to the polar opposites of the lexicon.

To get a sense of the effect the modified objective has, I trained the modified word2vec using tweets collected described in the experiments section and chose a single positive word “love” and a single negative word “hate” and evaluated on a set test words I considered to be synonyms of love and hate. The following histograms show the semantic orientation scores with the modified objective and without using the modification. The test words evaluated on modified word2vec has a larger spread than without using the modification:





Experiments

In all experiments, the goal is to predict the daily close price of the S&P 500. First, tweets are collected. Next, word embeddings are generated using the modified word2vec model and aggregate Semantic Orientation is computed. Finally, the daily SO values are used as features to build a regressor and classifier to predict the S&P 500’s close price.

First, I gather tweets and download a financial lexicon that has polarized words. I used the Twitter API to gather tweets that reference the S&P 500. I collected tweets that have one or more of the following keywords:

"S&P 500", "SANDP 500", "SP500", "#SPX", "$SPX"

The tweets that have the timestamp of being posted between the open of the previous day and the open of the day in question are used to make a prediction of the close price. The tweets are preprocessed using the same procedure outlined by Makrehchi et al. [4]. Each tweet is normalized by changing every link to become *LINK*, every account name to become *ACCOUNT*, stopwords are removed, words are lowercased, and each tweet gets tokenized using NLTK's tweet tokenizer.

In addition to tweets that reference S&P 500, tweets that reference the top 10 largest cap stocks of the S&P 500 have been collected. Similar keywords were used to collect those tweets. 907,766 tweets were collected between 04-19-2016 and 5-27-2016. 378,329 tweets are unique. The remaining tweets are retweets.

To compute Semantic Orientation, word embeddings are generated. Only the unique tweets were used to train the word2vec model. The tweets are composed of 5,991,907 words, in which 34,030 unique and frequent (>5 occurrences) words are used to generate the vocabulary. In addition to the tweets a stock market opinion lexicon was downloaded and incorporated in the model to be used to compute modified word2vec objective.

N. Oliveira et al. [5] created an opinion lexicon generated from stock market conversations on StockTwits, a microblog for stock traders. It was created by observing n-gram statistics of 350,000 posts labeled as bullish or bearish. Their lexicon consists of approximately 6,500 unigrams. Each word has a positive or negative score to indicate it’s sentiment. The positive words formed one group and the negative words as the other group.

In addition to the stock opinion lexicon (SOL) to build a word2vec model that captures stock market sentiment, 5 more word embeddings models were generated to capture sentiment on social issues. The reasoning behind considering social issues is that sentiment concerning social issues may also be a factor stock trading decisions.

The 5 social issues considered are employment opportunities, freedom from discrimination, quality education, honest and responsive government, and political freedom. For each social issue, a manually created polarized lexicon of approximately 15 "positive" and 15 "negative" words was chosen for each issue.

TensorFlow was used to generate the modified word2vec model. I used their skip-gram wordvec implementation and added the second loss term. The second loss function updates the weights after the first loss, effectively performing two weight updates each step. TensorFlow

allows for users to write an expression for an objective and its' optimizer computes the derivatives to perform backpropagation. The model performs stochastic gradient descent where it updates the weight after each example is evaluated. For all experiments I set the training to stop after 30 epochs and the word embeddings vector size is 200.

After the word embeddings are generated, an aggregate sentiment score is generated. I use scikit-learn's standard scalar to scale all scores across days to unit variance.

Evaluation

For evaluation, the dataset is divided into a 33% training set, 17% validation set, and 50% test set. First, the model selection is performed by training on the training set and performance evaluated on the validation set. For this part, I built and evaluated a linear regression model, SVM regressor with RBF kernel, and AdaBoost regressor. I found that the SVM regressor outperformed the the other models. Next, I performed the same process, but for classification. I evaluated a logistic regression classifier, SVM with RBF kernel classifier, and an AdaBoost classifier. The validation set revealed that logistic regression was the top performer. As a result, I use only logistic regression for classifying the test set and SVM regression for regression.

For the test set, I use a sliding window for training and testing. For each day, the training set grows by 1, the classifier is trained over again, and the next date tested. The results reported is the mean performance over all test dates. Accuracy is used to measure the classification performance. RMSE is used to measure the regression performance.

Different datasets were generated to see the effects of semantic orientation. The following were generated:

1) SOL: A single feature that represents the aggregate semantic orientation trained on the stock opinion lexicon (SOL) of all words tweeted on a given day.
2) POS SOL: Same as the first dataset, but the words that have the same part-of-speech are grouped together. I use the Penn treebank that generates 45 features.
3) SOL + SOCIAL: The stock opinion lexicon based SO plus SOs based on social issues. This generates 6 features.
4) POS SOL + POS SOCIAL: Stock opinion lexicon SO plus Social Issue SOs organized by their part-of-speech. This generates 270 features.

Each dataset was tested with and without sentiment of the large-cap stock tweets considered. The following table show the performance of the different runs:

Mean RMSE: Regression using SVR with RBF kernel

	SOL	POS SOL	SOL + SOCIAL	POS SOL + POS SOCIAL
Large-Cap tweets excluded	12.42	12.15	12.15	12.08
Large-Cap tweets included	12.03	12.01	12.22	11.90

Mean Accuracy: Classification using Logistic Regression L2 norm				
	SOL	POS SOL	SOL + SOCIAL	POS SOL + POS SOCIAL
Large-Cap tweets excluded	.46	.62	.38	.54
Large-Cap tweets included	.46	.54	.38	.69

The results show that best dataset used for both regression and classification is the POS SOL + POS SOCIAL. It's clear that dividing the sentiment by the words' part-of-speech improve performance. Also, incorporating the sentiment on social issues affect the model's predictability.

Next, I compare how the best regression and classification run compares to calculating SO using PMI and calculating SO using the same word embedding model that does not have the second objective. For each run, the features used is POS SO + SOCIAL SO, the classifier used is logistic regression and the regressor is RBF SVR.

Performance comparison with alternative approaches (Large-Cap tweets included)			
	Classification (Mean Acc)	Generated wealth	Regression (mean RMSE)
SO based on modified word2vec	.69	5	11.90
SO based on the unmodified word2vec	.62	3	11.91

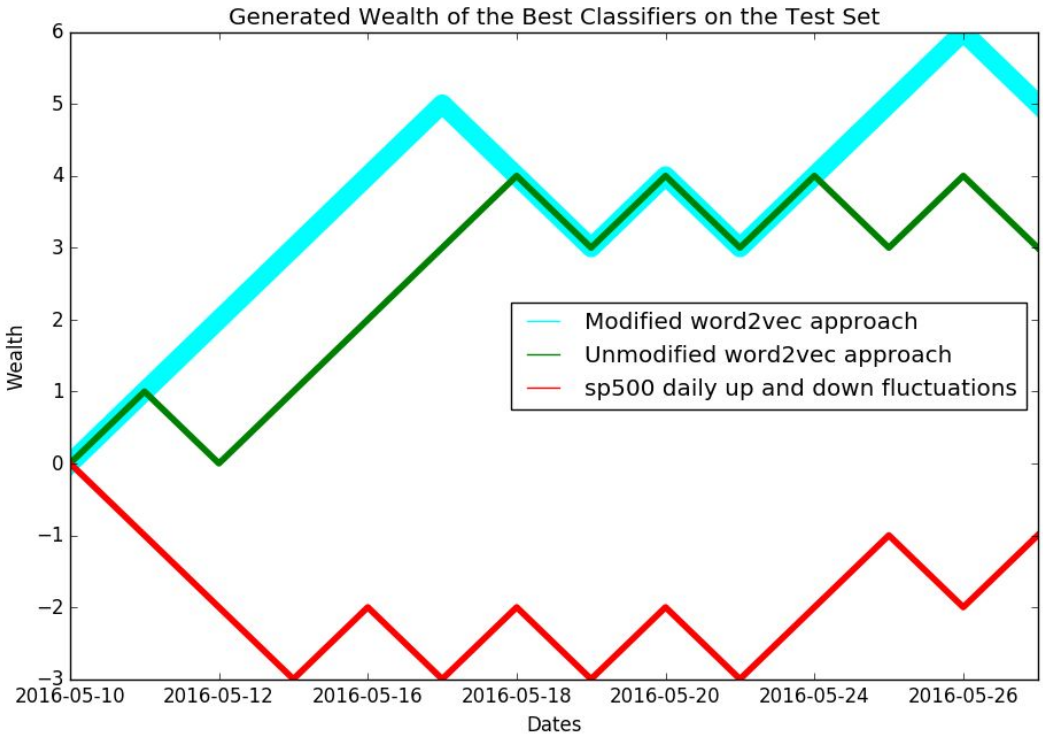
SO based on PMI	.54	1	11.94
5 day moving average	0.62	3	15.26
10 day moving average	0.62	3	16.23

Next, I run a t-test comparison of the accuracies of the three baseline approaches. The following table shows the p-values of the paired t-test:

P-value of paired t-test of algorithm classification accuracy			
	Classifier based on modified word2vec	Classifier based on the unmodified word2vec	Classifier based on PMI
Classifier based on modified word2vec	NA	0.58	0.43
Classifier based on the unmodified word2vec	0.58	NA	0.72
Classifier based on PMI	0.43	0.72	NA

The p-values of the t-test are all around the 50% mark. So, that suggests there is approximately a 50% likelihood that the performance of the three models cannot distinguished from one another.

Finally, in order to get a sense of how well these models would do in practice, I define the term wealth for the binary classification. The binary classifier can be used to purchase binary options where an investor makes a decision as to whether a stock will go up or down by a specified future time. Generated wealth is the sum of true positives and true negatives subtracted by sum of false positives and negatives. The following is the generated wealth of the best classifier:



The accumulated wealth by the last day for the binary classification was 5. It can be seen that the modified word2vec approach outperformed word2vec and accumulated more wealth than the S&P 500 for that time period.

The results show that all models outperform the S&P 500 for that time period. They also outperform the moving average indicators.

Previous experiments

Before running this experiment utilizes Semantic Orientation to predict the S&P 500, there were previous experiments ran. Previously, sentiment was used to make predictions. Tweets were gathered from users who regularly tweet predictions of the S&P 500. The tweets were organized in a dataset such that each row is day and each feature is a user that tweeted about the S&P 500. 61,521 tweets were collected from 2906 users from 10/18/2010 to 2/18/2016. This larger time span required 1305 daily predictions.

Each tweet collected was assigned a sentiment classification. In this experiment, I used the free api, Sentiment140.com to determine if a tweet had positive, negative, or neutral sentiment. The algorithm the api used is based on the paper, "Twitter Sentiment Classification using Distant Supervision" [10]. The researchers trained a maximum entropy classifier on tweets that contain smiley and sad face emoticons. The emoticons were taken out of the tweet and served as sentiment labels.

Using that model, predictions were made using the same machine learning algorithms. Even though the data was sparse, the best model still outperformed the baseline with an average binary classification accuracy of 57%.

Conclusion

In the future, it would be interesting to predict more future dates to see what effect it has on the algorithm comparison paired t-test. Also, I would like to include more polarized words of the social issues. I would expect more words in a polarized lexicon would improve the prediction performance.

In conclusion, this paper has shown that the sentiment of tweets be captured for stock market prediction. Twitter can be used a source of data for an intelligent trading systems. Using Semantic Orientation can be an effective feature for the task of predicting the daily rise or fall of the S&P 500. All models that use SO outperform the 5 and 10 day moving average and generate more wealth than holding the S&P 500 index. This paper also demonstrated that modifying the loss function of word2vec improves the prediction performance with an average accuracy of 69%.

Bibliography

[1] Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang. "Forecasting stock market movement direction with support vector machine." *Computers & Operations Research* 32.10 (2005): 2513-2522.

[2] Turney, Peter D. "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews." *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002.

[3] Nassirtoussi, Arman Khadjeh, et al. "Text mining for market prediction: A systematic review." Expert Systems with Applications 41.16 (2014): 7653-7670.

[4] Makrehchi, Masoud, Shalin Shah, and Wenhui Liao. "Stock prediction using event-based sentiment analysis." Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on. Vol. 1. IEEE, 2013.

[5] N. Oliveira, P. Cortez, N. Areal. Stock market sentiment lexicon acquisition using microblogging data and statistical measures. In Decision Support Systems, Elsevier, In press, 2016.

[6] Pujara, Jay. "Lecture 13 - Mining Discrete and Continuous Sequences" TIM245 course. UCSC Santa Cruz, CA. 26 February 2016.

[7] Vector Representations of Words. r0.8. TensorFlow.org. 20 May 2016.

[8] Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Vol. 1631. 2013.

[9] Ono, Masataka, Makoto Miwa, and Yutaka Sasaki. "Word embedding-based antonym detection using thesauri and distributional information." *Proc. of NAACL*. 2015.

[10] Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." *CS224N Project Report, Stanford* 1 (2009): 12.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, Advances in Neural Information Processing Sy

[12] Bondt, Werner FM, and Richard Thaler. "Does the stock market overreact?."*The Journal of finance* 40.3 (1985): 793-805.

[13] Tetlock, Paul C., M. A. Y. T. A. L. SAAR-TSECHANSKY, and Sofus Macskassy. "More than words: Quantifying language to measure firms' fundamentals." *The Journal of Finance* 63.3 (2008): 1437-1467.

[14] Tumasjan, Andranik, et al. "Predicting elections with twitter: What 140 characters reveal about political sentiment." *ICWSM* 10 (2010): 178-185.

[15] Renult, Thomas. "Can Twitter Help You Beat the Stock Market." The Conversation 1/1/16. Online.

[16] Vu, Tien-Thanh, et al. "An experiment in integrating sentiment features for tech stock prediction in twitter." (2012): 23-38.