# Buzzword Trading

A Sentiment Analysis Engine for Stock Trend Prediction


John Skandalakis
College of Computing
Georgia Institute of Technology
johnskan@gatech.edu

Sanya Ralhan
College of Computing
Georgia Institute of Technology
sralhan@gatech.edu

Fuad Hasbun
College of Computing
Georgia Institute of Technology
fhasbun3@gatech.edu

Daniel Ocano
College of Computing
Georgia Institute of Technology
docano3@gatech.edu



**Abstract -** ***Stock markets are highly volatile, traders at Wall Street etc. generally follow trends second by second to make money on that time granularity. Any ways to predict markets by understanding the trend hourly or more frequently can be really beneficial to such traders. In our project we aimed to study and determine if sentiment analysis of publically available social media feeds (specially Twitter ) can be useful to predict stock market trends. We created several experiments for controlled cases and found extremely positive results in some cases and also found cases where the results were not promising enough. We use supervised machine learning classifiers to get sentiment analysis of feeds and show results of our system. We conclude by stating that for certain major events there is indeed a strong correlation between the sentiment and stock prices but there are many other aspects to consider to build a foolproof prediction system.***


# 1. Introduction

There is an active line of research amongst traders to use powerful computers and processing to speed read news reports, articles, blogs and social media posts with the goal to build a automatic stock prediction system which finds opportunities for bigger gains and recommends investment suggestions to get best results.[1] A very important aspect of stock markets is the market sentiment and perception of how a company is doing in general. Therefore sentiment analysis of social media posts regarding companies is a logical and useful signal for trading. Our system can easily act as the heart of a full fledged recommender system mentioned above. We show that such systems are feasible and can indeed be profitable when used correctly.

The state of the art in sentiment analysis suggests there are 6 important mood states that enable the prediction of mood in the general public. The prediction of mood uses the sentiment word lists obtained in various sources where general state of mood can be found using such ‘bag of words’ or emotion tokens. With the high number of tweets posted on Twitter, it is believed that the general state of mood can be predicted with certain statistical significance.

For the purpose of this study, we will focus on sentiment, rather than mood. Sentiment is great, because it reduces the number of variables to work with in classifying stock option. In this paper, we focus less on stock option and instead focus on how twitter activity (and social media activity in general) affects the volatility of the stock market.

Our program, BuzzTrading, generates the sentiment data needed to study events in the stock market and we will use this data in the future to model stock options and when it is appropriate to buy them.

## 2. Motivation

Previous work has been a really strong motivation in our project. In fact originally our proposed methodology and approach for this project was a little different, we thought of building a news website crawler to classify sentiments on them and build sentiment analysis on that. More research into related works such as [2],[5]show that the sentiment analysis for news and blogs is much different from sentiment analysis on product reviews, there is a lot of noise. Moreover to crawl every news and blog site will take a lot of time and results are more skewed.

Instead in paper [3] the authors discuss algorithms to classify Twitter messages as positive or negative term by using machine learning algorithms. The authors mentioned the general goal of sentiment analysis for consumer research, product / service or market opinion collection. It was concluded that machine learning techniques perform reasonably well for classifying sentiment in tweets. In [4] they show how Twitter sentiment analysis is significantly different from similar analysis on news, blogs and reviews. This is due to the very small 140 character limit as well as the high usage of non-standard language and jargons and argots.

In [6] they use Efficient Markets Hypothesis to predict movements in stocks. They used SVM using frequency of words as feature to get a good performance system for 3-5 days range selection. We also know about startups such as [7],[8] that work on text analysis using social media Artificial Intelligence to make profitable business models.

Although there is a lot of work in this field as shown above but our project has many novel aspects including the approach, control cases, analysis, machine learning algorithms etc. Through our work we highlight when will it be more profitable to use sentiment analysis and when not.

## 3. Background

Predicting the Stock market has caught the eye of researchers everywhere. It doesn't matter what field you are in, there is always fresh data which can be used to make predictions on how the stocks are going to do.

Researchers at stanford were able to use machine learning to build a model which helps predict stock the stock market[16][18]. Their features come from the stock market itself, but we believe that adding external features, such as social media sentiment, will greatly benefit the process.

Bryce Taylor [17] uses 12000 financial analyst headlines to predict the stocks for 7 Fortune companies. His work is far from complete, but it does better than random chance. It should be noted that although this method is not perfect, it could be used in conjunction with something else to help improve the chances.

## 4. Methodology

In this section we discuss the methodology, including how we obtain information from Twitter, the stock market, and other social media sites. We also discuss how we classify the data and process it in a meaningful way.

### 4.1 - Twitter API versus Workaround

There were two solutions to obtaining data from Twitter. Use the Twitter API, or use the primary Twitter endpoint. Both solutions have setbacks and perks.

Twitter API's main perks are that it provides the data quickly and responsively and it makes it easy to obtain a live stream of data. This is the right solution for analyzing the stock market in real time.

The setbacks are the it is costly and that it is really bad at getting historical data. Twitter does not provide old tweets in chronological order and they remove many tweets from queries. We really needed all of the old tweets, so we decided to use a workaround from Jefferson Henrique [15]. Another downside is that Twitter does not allow getting all tweets from between a range of time.

Jefferson's method requests tweets from Twitter's main page and in the request header, specifies we want the data in a JSON format. Then it aggregates all of the results within the desired time period. You can also specify a search query using this method, which is extremely helpful when searching for companies. Twitter's built in search function is a treasure cove for finding data pertaining to any particular subject, in our case it is publicly traded companies

The downside of this method, is that it is slow. It takes between 5 and 15 seconds to get 100 tweets, and it does this until you have all of the necessary tweets. Getting a week's worth of data for a topic that is extremely popular, can take a few hours to run. To upside is that it is free, it shows data from all users, and it is sorted by date.

## 4.2 - Stock Market Data

The fluctuations in the stock market are relevant for our sentiment analysis as we aim to find a correlation between our analysis and the stock market. At the moment, the stock market data is not directly integrated with our system. Instead, we first have to run our sentiment analysis on a given set of data and afterwards use the obtained results to compare them with the stock market data manually. To do so, we have developed a set of functions in python that help us examine our outputs. Such set of functions are denoted in **Figure 1** and were developed by using a python wrapper for yahoo's stock market data.

**Figure 1**

```
#getYahooStock provides the stock value of a
#company for two specific dates
#@param ticker the ticker of the company
#@param date1 first date for desired stock
#@param date2 second date for desired stock
def getYahooStock(ticker, date1, date2):

#stockDrop will return the first stock that
#drops below the stock value at a given date
#@param ticker the ticker of the company
#@param date1 date for original stock
def stockDrop(ticker, date1):

#buyStock runs a simulation of buying and
#selling stocks. It lets you buy an amount of
#stock for a specific company and will output
#the value of #your stock after provided date
#@param ticker the ticker of the company
#@param buyDate date when stock is bought
#@param sellDate date when stock is sold
#@param amount total stock bought at buyDate
def buyStock(ticker, buyDate, sellDate, amount):
```

Additionally, we also used Google's Finance graphical user interface in order to analyze our results after running the sentiment analysis. More specifically, we used the Stock Price vs Time graph that displays the stocks of a given company against a timeline in order to understand the changes in stocks between a given set of dates.

## 4.3 - Other Media

There is other media worth noting, which could add elements to our final equation. We are not here yet, but we hypothesize that some of the following media outlets could contain impactful sentiment.

*Reddit* adds a layer of filtering, which could be used in junction with Twitter to figure out which tweets are more important. Tweets tend to indiscriminately get like just because someone is famous, but Reddit could narrow down the cases where a less important person makes an impactful tweet because it will get upvoted on Reddit by many people who don't use Twitter.

Originally, we had intended to use MSM (mainstream media) as a source for our sentiment, but crawling it and parsing it is expensive and takes time. Other work we have done includes a distributed crawler, using scrapy which can crawl about 1000 pages a minutes. The main caveat is that if you want to get all of the historical data, this will take a lot of equipment to cover in a reasonable amount of time. In addition, these site might block you after crawling too much for an extended period of time. Waiting 3-5 seconds between articles may be necessary. Articles also have a lot of text to parse through, which means creating the vector will be very taxing on the system.

In addition to these technical setbacks, there is the question of which sentiment is more impactful? We believe that social media sentiment is less artificial and demonstrates a more honest view of the market sentiment. Since MSM selects their editors, writers, and journalists, this means that they are likely biased in the sentiment that they do use. This doesn't mean there isn't an impact on the market, it just means that the impact may not be as strong as social media.

## 4.4 - Classification

Here we discuss all of the detail that went into building our model for classifying tweets as positive or negative. In the future, we plan on building additional models which take the output of this along with other variables to rank which stock option you should buy for a given company. Currently, we have a model which predicts the sentiment of tweets, and the output is helpfully when trying to make sense of the stock market.

**Train Data** - We used a dataset of 1.6 million tweets, which are classified as positive or negative. We obtained the set from ThinkNook[14] and it is based off of data from the University of Michigan and Niek Sanders. The tweets in the dataset are all general tweets that aren't about any particular event, as to avoid words like company names. Obviously, some company names cannot be avoided because they are based off of common words, or because people use the company name as a common word.

**Vectorizer** - As for our vectorizer, we took a bag of words approach. We gathered all of the words from all of the tweets and ranked them by importance. We did this using term frequency–inverse document frequency (TF-IDF), which numerically assigns an importance to each word in the data set. Because the data is classified as either positive or negative, words in positive tweets will be assigned a negative association or a positive association because of how frequently they appear. If words appear very frequently, then they will be dropped automatically because they are less significant. Such words are commonly known as "Stop Words".

**Classifier** - We tested our program with three classifiers which are commonly used for this type of data. We had the greatest success with LinearSVC. In comparison with SVM classifiers that used 2 different kernels; an RBF Gaussian kernel and a linear kernel, the Linear SVC classifier outdid both. Though using an rbf kernel is explicitly better than a linear kernel (The linear kernel has been proven to be a degenerate form of the rbf kernel), finding the

optimal conditions and tuning the kernel's options is difficult. The reason why we decided to stick with linear in the face of this fact is due to linear classifiers' speed and their accuracy when dealing with large amounts of data like the one we used. When comparing LinearSVC to a SVM with a linear kernel the LinearSVC performs better due to transformation costs. Since the SVM is flexible with what kernel it uses it needs to perform different transformations based on the input form for that kernel. LinearSVC worked easier and ran faster for the number of examples we wanted to use.

**Table 1 - Classifiers**

| classifier | precision | recall | f1-score |
|---|---|---|---|
| Linear SVC | 0.80 | 0.80 | 0.80 |
| | | | |
| | | | |

**Table 2 - Linear SVC**

| classifier | precision | recall | f1-score |
|---|---|---|---|
| Negative | 0.80 | 0.79 | 0.79 |
| Positive | 0.79 | 0.81 | 0.80 |
| Average | 0.80 | 0.80 | 0.80 |

## 4.5 - Architecture

Buzz Trading checks to see if a cached vectorizer exists, if not, it generates it using the training data. If it does, it simply loads it to memory. Then it does the same thing with the classifier. After this, it aggregates through all of the relevant tweets. Working with 100 tweets at a time, it fits and predicts the tweets. An OrderedDict is used to maintain the time intervals tweets are collected as the key, and inside is stored the number of positive tweets and the number of negative tweets at that given period of time. Once all of the tweets have been collected, the csv is written and several graphs are generated and saved. After this is finished, the console will print out the accuracy and precision for the classifier that was used in this test.

## 4.6 - Input/Output

We designed the program to have a simple input. All data required is cached or it gets it from the internet. There is a file called SAD.csv which contains A start time and an endtime is required for input. Optional inputs are the Twitter user and the Twitter search query.

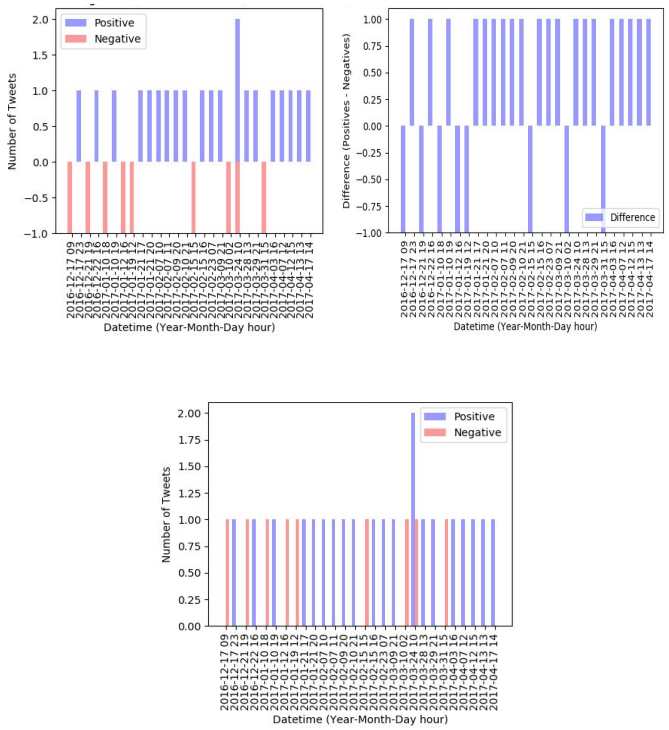
At this current time, you cannot adjust the granularity of the timestamps. By this, I mean you cannot adjust the period of time in which tweets are collected, but we plan to add this function in future stages of development.

The output comes in two forms, a csv and graphs, saved as PNG. The CSV is necessary in case we want to further analyze the data and the graphs are not enough. The following table is what the header data looks like and the figure 1 shows what standard graphs are generated from this software. We also give users the option to print out information on the machine learning portion if they

**Table 3 - CSV header**

| TimeStamp | Positive Tweets | Negative Tweets |
|---|---|---|

**Figure 2 - Output Graphs**




Histograph of Positive and Negative above and below (Top left), historgram of Positive-Negative difference (Top Right), Histogram of Positive and Negative, side by side (Bottom)

We plan to add a live view for queries that updates every few minutes. This shouldn't be difficult to implement as it will rely on the same underlying model and architecture. The primary difference will be that it gets data from the Twitter API instead of from the twitter search endpoint.

# 5. Performance

Here we will discuss our system information, including hardware and OS and we will discuss the required system resources for our software to run properly.

## 5.1 - System information

We ran our code on a Windows 10 machine (Version 10.0.14393 Build 14393) using the Ubuntu bash shell (Ubuntu 14.04.4 LTS). Our machine has an AMD-6300 Processor with 3 cores and 6 logical processors. We used a 480GB SSD and 16GB 1600Mhz RAM. Graphics probably did not make a huge impact on this project, but it could if you use the version of Numpy and Scipy[12] which take advantage of GPU for calculations. We used a NVIDIA GTX 980. Overall, the system costs about $700.

We ran this on a home network with 300 Mbps download and 30 Mbps upload speed and a ping of around 11ms, provided by Comcast. We ran this over a wireless airport extreme router.

## 5.2 - System Resources

The cached vector uses about 667 Mb of disk space and we expect it to use that amount of memory. Even more memory is used when creating the vector, because of the data processed by the vectorizer. This primarily includes the training data which is 150 Mb.

The LinearSVC Classifier only used about 3.2 Mb of disk space when cached and we also expect it to use the same amount of memory. A small amount of memory is used to fit new data for every 100 tweets we gather. This is short lived. Overall, we don't expect BuzzTrading to ever use more than 2Gb of Memory.

As for network usage, we expect our program to slow down on systems with download speeds less than 5 Mbps. Every 15 seconds, the program downloads data from Twitter reaching a peak of 4.1 Mbps. This operation lasts for about 2 seconds. We could improve the 15 second delay by

making request over multiple VPNs concurrently, enabling us to bypass Twitter's rate limits.

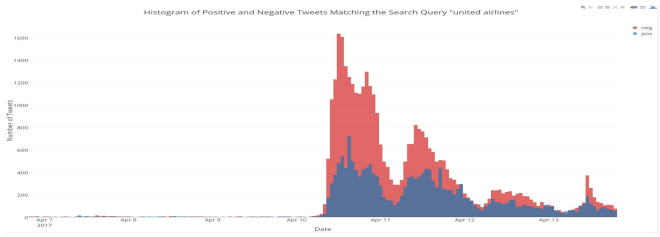
At any given time, while running BuzzTrading, we never used more than 33% of our CPU usage. Most of the time was reading and writing to disk. With an RBF classifier, we used more CPU, but this was less accurate as well.

Our primary bottleneck in this experiment, was nothing on our system. We constantly remained under our system specifications. The primary thing which affected the speed of this program was Twitter because of how often we could make a request. Twitter has a rate limit and it is important that you don't exceed the limit or Twitter will not respond with the data you want. If we use the VPN option to bypass this, we would expect CPU to be the main bottleneck.

# 6. Analysis

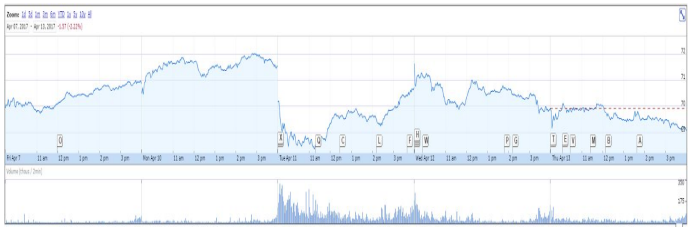
In this section, we will discuss our findings, including where our product excels and it's shortcomings. It works particularly well when someone influential tweets about a company and when there is general public outrage. It tends to fail with companies that people are always tweeting about. It also tends to fail when companies have non unique names, where the name of the company is used regularly in a different context. Like the word "Google" refers to the company, but people also use it regularly in such a way that has nothing to do with the company's value.

**Figure 3 - "United Airlines" query sentiment**




Positive (Blue) and Negative (Red) Twitter sentiment for the query "united airlines" in Twitter search results between the dates April 07 and April 13.

**Figure 4 - UAL Stock Ticker**



United Airlines (UAL) stock ticker between April 07 and April 13, courtesy of Google Finance GUI.

A use case that worked particularly well is the recent United Airlines incident, where United called on security to forcibly remove a passenger from a plane to make way for United employees. The result was public outcry over a company that is rarely ever mentioned in social media. As seen in figure x, the term "united airlines" gets no more than 20 tweets per hour up until April 10. Around 9 AM, a hundred people are tweeting. By 10, it's hundreds and by Noon it's over 1000 per hour. The sentiment is overall Negative. The stock market closed at 4PM and reopened the next day significantly lower. The initial burst of negative tweets seem to indicate that the stocks are going to take a hit. The residual negative sentiment seems to indicate an increase in volatility, the UAL stocks look less stable than they did the previous days.

# 7. Future Work

Our plans for the future are to create a model which combines this data from Twitter with other variables from the stock market. With enough data, we believe we can create a system which can intelligently suggest which stock options to use for a given company. In order to to this, our program will need several components.

A component which tells the user if our model will not work on a specific company is necessary, because not all companies look the same when using Twitter data. It is easier to change which companies we analyze than it is to change the algorithm to work on every company. Basically, this feature will require analysing the name of the company and the relevance of Twitter results returned. It will also need to analyze historic data from that company and see if Twitter sentiment is even a good feature because not all companies are equal in this respect.

A component which shows live Twitter data side by side with live stock market data is necessary for visually analysing the data so we can come up with a model for making stock option suggestions. MatPlotLib[11], which we used to graph our data, has a feature called ion which will update the graph whenever called. It will be interesting to see what the stocks do as stuff happens on Twitter. One thing we expect to see is that not only will Twitter sentiment affect the stocks, but in some cases the stocks may affect Twitter sentiment. The is something we could observe with a side by side view.

Finally, we also need a component which makes the actual stock option suggestions. This will be the most challenging portion, but will pay off if successful. We can do this by creating models for each stock option. The features will include, but not be limited to our sentiment score, volume of tweets for that company query search, and external data which is known to affect the stock market (like candlestick patterns)

# 8. Conclusion

We conclude by stating that social media sentiments do have a strong correlation with the image of a company and their stocks . In special cases it may be more relevant than other and hence systems should incorporate that. With machine learning research in place it may be possible to accurately analyse social media sentiment as well. In the future it may be useful to research into incorporating other factors such as company profits, earnings etc alongwith sentiment analysis data from a wide range of data sources to build recommender systems that will predict stock predictions accurately much better than humans.

# References


[1] Graham Bowley, ”Wall Street Computers Read the News, and Trade on It”, New York Time

[2]Large-Scale Sentiment Analysis for News and Blogs,http://icwsm.org/papers/3--Godbole-Srinivasaiah-Skiena.pdf

[3]Alec Go, Lei Huang and Richa Bhayani, ”Twitter Sentiment Analysis”, CS224N Final Report, 2009.

[4]Ravi Parikh and Matin Movassate, ”Sentiment Analysis of User-Generated Twitter Updates using Various Classification Techniques”, CS224N Final Report, 2009.

[5]Ryan Timmons and Kari Lee, ”Predicting the stock market with news articles”, CS224n Final Report, 2007.

[6]Gabriel Fung, Jeffrey Yu and Hongjun Lu, ”The Predicting Power of Textual Information on Financial Markets”, IEEE Intelligent Informatics Bulletin, Vol. 5. No. 1, June 2005

[7]Isentium,http://isentium.com/

[8]Theysay,http://www.theysay.io/

[9] P. Ercolano, "Social media sentiment offers clues to stock performances, study suggests," The Hub, 2016. [Online].

[10] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011. [Online].

[11] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[12] "The NumPy array: A structure for efficient numerical computation - IEEE Xplore document," 2017. [Online].

[13] D. Myers and J. W. McGuffee, "Choosing Scrapy," *Journal of Computing Sciences in Colleges*, vol. 31, no. 1, pp. 83–89, Jan. 2015. [Online].

[14] "Twitter Sentiment Analysis Training Corpus (Dataset)", ThinkNook, 2017. [Online]. Available: http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/. [Accessed: 24- Apr- 2017].

[15] Jefferson Henrique, “Get Old Tweets,” https://github.com/Jefferson-Henrique/GetOldTweets-java

[16] Shunrong Shen, Haomiao Jiang, Tongda Zhang, “Stock Market Forecasting Using Machine Learning Algorithms”

[17]Bryce Taylor, “Stock Market Prediction Using Artificial Neural Networks”

[18] Prakash Ramani, Dr. P.D. Murarka, “Stock Market Prediction Using Artificial Neural Network,” *International Journal of Advanced Research in Computer Science and Software Engineering,* Volume 3, Issue 4, April 2013