Socci Trading: Predicting Stock Market Behaviour using Twitter Sentiment

Systems Design Engineering

University of Waterloo

SYDE 461

Jeyavithushan Jeyaloganathan 20348912

Harsh Dave 20343950

Senduran Indrakumar 20333636

Ziyad Mir 20333385

Advisor: John S. Zelek PhD, PEng,

Graduate Chair of Systems Design Engineering

Abstract

In this report, we outline a stock prediction model for an application that will leverage public Twitter feeds to predict stock fluctuations of several companies. Historically, stock market movements have been highly unpredictable. With the advent of social networks and new machine learning and natural language processing techniques, financial institutions and researchers have developed computerized mathematical models to maximize their returns while minimizing their risk. However, there is an area that has been unexplored in predicting stock market behaviour. *As emotion is a key influence on the stock market, can we analyze social signals to determine the general market sentiment and leverage this in machine learning models to predict how the stock market will behave?* We propose an approach to understand the market sentiment – its mood – in relation to several companies and subsequently use this sentiment to predict market behaviour and identify opportunities for investment. We do this by building an application that will determine the public sentiment using natural language processing techniques such as word lists and then feed this sentiment into a support vector machine and linear regression model to ultimately determine an appropriate investment strategy to maximize profits.

Table of Contents

Table of Figures and Tables

Overview

Historically, trading has been done through human intuition and the analysis of market trends. However, with the advent of the modern computer, there has been a paradigm shift. Today, algorithmic trading dominates the financial trading industry.

In 2004, Facebook was launched; it started the social movement which resulted in a new type of ubiquitous information - social data. These social signals describe not only the events and activities of people and groups but also their mindset and emotional state. Social networks such as Twitter, Facebook, Quora and Google+ to name a few, provide a powerful source of insight into the mindset of people. And it is people, their expectations, behaviour and their sentiment that serve as one of the primary influencers of the stock market [1, 2].

We believe that we can leverage the staggering amount of social data publicly available to provide powerful analysis, deep insight, and meaningful predictions for trading. Initially, we intend to develop a prototype that will scrape a few keys social networks such as Twitter, Facebook, Google+ and Quora for keywords and analyze them to decipher trends and make meaningful predictions on how the stock market will move. We will use both offline and online machine learning techniques to create a prototype algorithm. We plan to train and test it using freely available, historical data. The initial scope will be constrained to the Fortune 500, and a few stock markets and indexes.

Ultimately our goal is to develop a model and application that will use social data to ascertain the mindset of the market and leverage this information to predict fluctuations in the stock market in order to make more profitable investments.

Background Information

Equity Trading

In finance, equity trading is the buying and selling of a publicly traded companies stock shares. To trade a company's stock shares means to allow for its selling or buying on the various stock exchanges around the world. Some of the most prominent stock exchanges are the New York Stock Exchange, NASDAQ, the Tokyo Stock Exchange, and the London Stock Exchange.
There are many variables that influence the trading price of the stock of a given company. One of the main variables is the perceived value of the company by the market. This is known as the *market*

sentiment. Typically, stock investors or traders will try to predict how the stock market will move based on certain clandestine information and/or other determining factors such as the market sentiment. Based on their prediction, investors will either buy or sell stock to increase their margin of profit. Fundamentally, there are two basic methods to earn profit through equity trading:

1. **Buy and Hold (B & H)**

This first method describes the act of buying stock at a certain price point, in the hopes that its price will increase while holding it. Subsequently, one can sell this stock at the higher price. The difference between the buying and selling price, and the quantity of stock bought will determine the profit. Traditionally, this has been a fairly low risk method for earning money as long as the right stock is bought (done through limited research). However, if you could predict how the stock will behave, then at a fairly low risk, you can make large amounts of money by buying stock that you *know* will increase and then selling it later a higher price.

2. **Shorting**

Shorting describes a technique for trading stock that will allow for profit when the price of a given stock for a company is declining. If there are indicators that a given stock will decline, then you can borrow this stock from another trader with a promise to pay back the total amount of stock borrowed. You then sell this stock at the current price (as it will go down) and then buy the same amount back at the lower price to return to whomever you borrowed from. Once again, if one can accurately predict how the stock market will behave, there is a large opportunity for profit.

A common determinant in the success and validity of these two methods is the ability to predict how the stock market will behave. As previously mentioned, a powerful indicator for this is the *market sentiment.* It describes the mindset of the market. Typically, with simplification, if the sentiment of the market is positive, then the stocks for that market will increase. If the sentiment is negative, then the price of the stock will decrease. You can also say that if the market sentiment is neutral, there will be little chance of large fluctuations in the market. The probability that a given stock price will experience large fluctuations is called volatility. The market sentiment can be analyzed to ascertain the volatility of the stock market as well as the direction it will move.

Social Signals

Today, Facebook has over 1 billion users and Twitter has over 500 million users with 58 million tweets a day, and 1 billion every 5 days [1, 2]. These are a few of the many social signals available to determine

market sentiment. These social signals describe how the general populace perceives companies which then influence the price of stock markets. However, social signals are only largely available for companies that occupy large mindshare. This means that social signals can only be used to determine the market sentiment of stock that the users that create these social signals are aware of and inclined to post about. The sheer magnitude of the amount of social data available for analysis is what will allow for a wide breadth of companies and stock to analyze.

Relevant social signals will contain some indicator correlating with stocks of interest as well as language describing its emotional message. For example "The new iPhone 5s! #iPhone." Analyzing this, we can determine that it is probably a positive statement about the company Apple. With advanced NLP techniques, we can determine the sentiment of this social signal as well as its subject.

Problem Statement

This leads to the problem we are trying to solve:

> ***How can we utilize social signals to assess market sentiment and subsequently predict the behaviour of the stock of certain companies, markets and stock indexes to identify rich opportunities for equity trading?***

Objective

In order to create a feasible system for identifying significant opportunities for equity trading based on user sentiment, we need to narrow down the scope to prove whether it is feasible. That being said, our prototype will leverage Twitter as our exclusive social signal and analyze it to determine the behaviour of the following five stocks: Google, Apple, Microsoft, Goldman Sachs, and the Standard & Poor's 500 index. Twitter was selected as our social signal because of its real time facet and the importance of real time decisions in equity trading. The five stocks were determined based on their visibility and mindshare on Twitter, ensuring that we will have enough data to model how their social signals affect their stock.

In summary, our objective is to analyze Twitter tweets to predict how the stocks of Google, Apple, Microsoft, Goldman Sachs, and the S&P 500 will behave to identify opportunities for equity trading.

State of Art

Regarding the state of art, there have many attempts and approaches to determine market sentiment to predict fluctuations in the stock market. This section will review the state of art and analyze their strengths and weaknesses.

Bollen et Al.

In 2010, Johan Bollen and his team published a paper describing how they analyzed tweets to determine its sentiment (calm, alert, sure, vital, kind, alert, and happy) and used it to predict the value of the Dow Jones Industrial Average (DJIA). Their solution compromising of a Granger causality analysis and a self-organizing Fuzzy Neural Network resulted in an accuracy of 86.7% for predicting the daily up and down changes in the closing values of the DJIA and a reduction of the Mean Average Percentage Error (MAPE) by more than 6%. The two most noticeable gaps in their approach was a lack of localization to the tweets they analyzed and missing ground truth to verify the veracity of their calculated market sentiment [3].

Chen et Lazer.

In 2011, Ray Chen and Marius Lazer outlined a modification on the Bollen et Al. algorithm to using Twitter sentiment to predict market fluctuations. The used a different approach to determining Twitter sentiment. They used a word list to evaluate emotions and averaged these per day to calculate the sentiment. They then employed two different investment strategies – classification, and regression. Based on their simulation, it was discovered that their simpler sentiment analysis technique was just as effective as Bollen's except for its advantage in time complexity and space complexity – it required less time and memory to run. Furthermore, a useful insight garnered from their research was that an effective timeframe for predicting the market behaviour was 3 days [4].

SNTMNT

SNTMNT is a start-up launched in 2012 that provides an API which can be used to monitor Twitter based sentiment to give advice on whether to buy or sell stock on an hourly basis. SNTMNT's trading indicator API gives price predictions based on Twitter sentiment about S&P 500 index stocks. Using machine learning algorithms, the company says that it has an accuracy as high as 60%, averaging at 54%. The API can produce hourly and daily buy and sell recommendations, with a 'confidence interval' to indicate how accurate the prediction is likely to be [5].

Goals and Requirements

Objectives

In our overview, we briefly discussed the problem and objective of our project. In this section, we expand on our objective and based on it, derive requirements. There are three main goals for our project:

1. Predict stock market trends by analyzing Twitter sentiment.

2. Provide a public interface that allows anybody to use our model.
3. Learn and investigate how machine learning techniques can be used to identify trends.

Translating these to concrete deliverables, our goals are to:

- Implement a machine learning and natural language processing (NLP) model to learn from millions of tweets and historical stock data to predict fluctuations in the market and subsequently identify opportunities for investment.
- Create an application interface to display recommendations on how much to invest in either buying and holding or shorting specific stock.

The main constraint on our problem is the limited data available to us through Twitter's Streaming API. We will only have access to a certain subset of the global tweets at any given time using the Streaming API. This creates an interesting caveat that needs to be taken into account when coming to conclusions on sentiment and giving recommendations. The second constraint is the amount of historical stock trading data available to us. The quality of our machine learning model depends directly on the quality. While these constraints are to be kept in mind during development, we are confident that they we can work within them and create an effective solution to our problem and achieve our objective.

Requirements

While researching the state of art and iterating on our problem and objective, a set of requirements were delineated to aid in feature development and project management. These requirements were gathered by analyzing the state of art, and translating our objective into quantitative requirement. Other requirements include functional, user experience/usability, data (input/output), and software component requirements.

Functional Requirements

The most important functional requirement is a robust, accurate, and reliable machine learned investment model. It should be able to perform better than the average prediction accuracy by other models: roughly 70%. This means, that our model should make accurate predictions 70% of the time. Additionally, the manner in which we display this should be concise and easy to understand.

User Experience and Usability Requirements

User experience and usability requirements include providing an intuitive and elegant interface for users, product support in the form of tooltips or mini-tutorials, and a clear product purpose. Navigation of the

application should be intuitive with a minimal learning curve and ease of use to encourage an engaging and interactive user experience.

Data Requirements

Requirements for system data are the ability to provide a filtered set of a minimum of 2 million public tweets containing names of listings as the primary input to our computation engine. To ensure that our system is kept up-to-date, all tweets to be processed must be time stamped. Ideally, the system will automatically update the database with the newest datasets facilitating real-time processing. When displaying appropriate investment strategies based on our machine learning engine, we must be able to populate the models and views in real time with minimal load delay in order to be considered dependable in the time-contingent investment industry.

Software Component Requirements

Software component requirements can be sectioned into storage and external dependencies. Storage requirements include local storage of the code repository as well database storage of at least 2 million tweets with metadata. While local storage is used for developing and testing, a web-based hosting service will also be required for deploying the application. External frameworks and libraries will be required to allow for rapid prototyping and data visualization. Database storage should hold a minimum 1 terabyte of data to ensure that our system can process all the tweets and their metadata without running out of space. For templating, an MVC framework should be used to ensure proper view generation and responsive design. It should also support JavaScript and HTML/CSS.

Algorithm Requirements

Our algorithm must be fast in order to be competitive. This is a must when working with massive amounts of data in real time, as is the case in the stock market. Our algorithm must also be accurate. Naturally, we needed an algorithm that would model the data as accurately as possible. Since our data is, by its nature, very noisy, we require a simple model to avoid high variance.

Design Constraints

As we progress closer to the final implementation of our project, there are a number of constraints that we continue to combat through all phases of the project.

Twitter API

The Twitter API limits the number of tweets that can be pulled in a given window of 15 minutes. Limiting the sample set can potentially skew the results we have since it may not be an accurate prediction of the market's movement. Currently, only 10% of all global tweets can be pulled. We work with the assumption that the 10% filters out any biases by virtue of random sampling, and that the other 90% of tweets we don't have access to will reflect roughly the same results that we obtain.

Natural Language Processing Quality

NLP in and of itself is a modern branch of computer science that is still in a stage of research, to some extent. Computers, at this point in time, cannot comprehend complicated linguistic phenomena such as sarcasm and colloquial words that may mean something else by the dictionary (cool, hip, sick). Combating this constraint entails ensuring that these words and phrases are filtered out and not used as a part of our algorithm to understand sentiment.

Limited to Large Companies

Twitter is a platform that is very conducive to the data analysis of the general public. For this reason, only companies that are very much in the public eye can be accurately analyzed without biases. Lesser mainstream, but just as impactful companies may not yield enough activity on Twitter to make definitive statements on the movement of the market.

Languages

Natural Language Processing involves the calculation of sentiment based on words used from a given language dictionary. This means that in order to incorporate support for languages in addition to English, some requisite knowledge of linguistic conventions in those languages is needed. The experience of the group requires that all of our algorithms be optimized for the English language. It's important to be aware of the fact that this renders a significant portion of global tweets irrelevant, since we will only filter by English. If perception of a company is highly negative in a country with a different first language, we will not be able to use their tweets to study sentiment.

Methodology

When considering our approach for designing our solution, we decided to employ a tiered strategy. This strategy consists of using our requirements and state of art to define important criteria for evaluation potential solutions. We will evaluate each component of our design, and then our global solutions according to the criteria. It is important to note however that the efficiency of our algorithm is the most important determining factor in the success of our project. Therefore, our final decision for our prototype will primarily rely on the evaluation metrics for the machine learning model, and natural language processing algorithms. Furthermore, we will employ an iterative design approach for developing our algorithm and application. This approach is well suited for machine learning problems as it already leverages a train-test-iterate methodology. Initially, we will propose three different solutions that could potentially resolve our problem. These solutions will be compared with one another with a set of criteria in mind and matched with the aforementioned evaluation metrics in order to arrive at the best possible design solution. Upon completing the selection of a solution, we will iterate on it as we move forward in order to keep the process fluid and in order to allow us to adapt based on our results.

Solution Criteria

The primary criteria we took into account were scalability, cost, algorithmic efficiency, and framework, respectively. Each of this was evaluated individually and systemically in order to come up with the best possible final proposed solution.

Scalability:

One of the most important criteria that we considered was scalability. How would our system respond to a growing set of data? The proposed solution had to have the ability to support the fact there would be a large amount of data being pulled from Twitter on a daily basis.

Deployment Cost:

Considering that this is a school project, deployment cost was definitely an important factor in choosing a proposed solution. Minimizing the monetary cost is important, but so is flexibility with languages and computational performance.

Algorithmic Efficiency:

As mentioned above in our requirement section, the algorithm we select will be motivated by several important factors:

Speed - Finding a fast and efficient algorithm was a primary concern for our project. When taking into account that the stock market has a huge amount of data and much of it needs to be interpreted real time, the efficiency of the algorithm becomes that much more important.
Accuracy - Stock market data and tweet data is very noisy, and the nature of machine learning algorithms is still very noisy. For this reason, we're looking for an algorithm that would model the data as accurately as possible.
Ease of Implementation - How well does the algorithm fit in with the rest of the system? If we use external libraries for complicated machine learning techniques or mathematical models, is the code in shipping condition and is it understandable?

Framework:

Familiarity with the web framework is an important consideration, as well. The amount of tedious work reduced by the framework from a templating and architecture perspective were important factors. Lightweight and flexible frameworks would certainly be considered beneficial for a project such as this one. How do the frameworks fit in with the preferred databases and deployment method?

Algorithm Design

In this section, we consider two alternatives for our algorithm as well as outline our final solution. The model for our project can be broken down into three sections: Sentiment Analysis which consists of natural language processing (NLP) algorithms, Machine Learning for understanding how sentiment can be used to predict stock behaviour and finally our investment strategy based on our machine learned model. The two alternatives we propose each use unique algorithms for the sentiment analysis and machine learning. The investment strategy is a common component and will be discussed in our final delineated algorithm.

First Alternative

For the first proposal, the core algorithm that will be employed by our system will utilize a custom machine learning model that leverages both Google Profile of Mood States (GPOM) and Particle Swarm Optimization (PSO). GPOM is a reliable natural language processing algorithm used by many of Google's complex software systems. It will be used as the component within our infrastructure for analyzing sentiment from the aggregated social signals i.e. Twitter tweets. PSO will be using the sentiment analysis gathered by GPOM in conjunction with historical stock market prices and trends to find and predict optima, in regards to various stock transactions i.e. buying, selling and shorting.

Second Alternative

For the second proposal, the algorithm that will be used combines both a Natural Language Processing (NLP) algorithm involving the "word list" technique and the Neural Network methodology. The "word list" technique leverages a comprehensive list of words (first composed by Alex Dave) that provide a sound representation of the sentiment spectrum, i.e. sentiment ranging from extreme sadness to pure happiness. This list can be used to find appropriate matches from the incoming social signals (tweets) with the use of pattern matching algorithms. With the use of the aggregated results gathered from the NLP stage, a neural network system will be applied to help predict confidence levels for stock market transactions. The application of a neural network revolves around the simplified model of the brain. Each layer of neurons are represented as a web of logical nodes (weights); the logic for these nodes consist of historic stock market statistics. Computations are performed in the interior layers on the "neuron" values create weights, which affect the next layer (a chaining effect). The entire network is composed of multiple layers, a top layer for input values consisting of sentiment values collected from the NLP layer, middle tier layers containing stock statistics and an output layer that provides final computed values, which represent the desired optima.

Final Algorithm

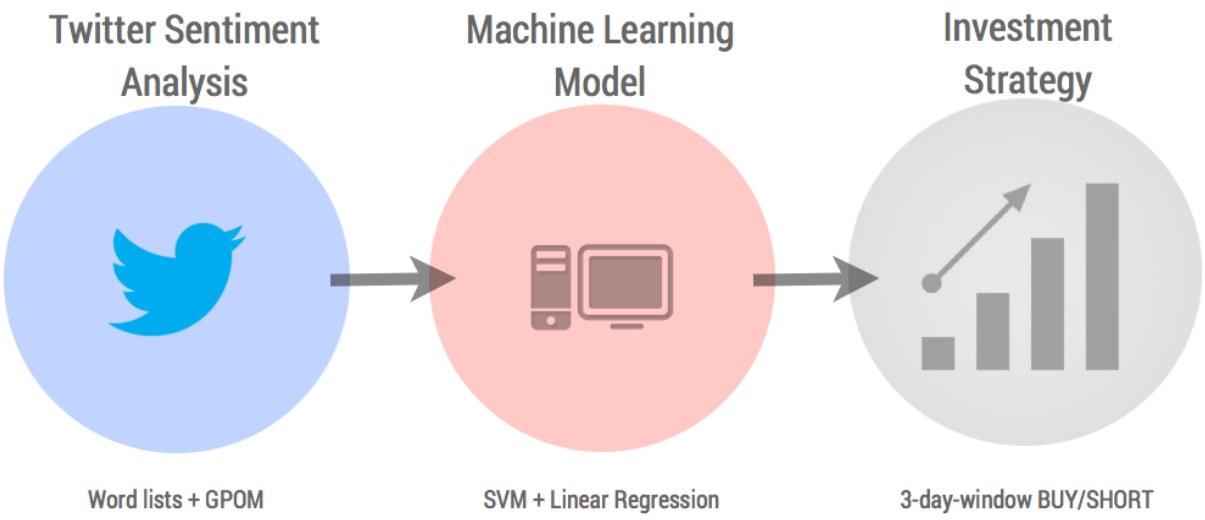

Figure 1 Final Algorithm

Our algorithm consists of three main components; sentiment analysis, a machine learned model, and an investment strategy. For the sentiment analysis, we decided to use the word list and tweet tokenization

method outlined above, in conjunction with GPOM. Our machine learned model will consist of three main features, trained and classified using Support Vector Machine and Linear Regression. We are using a hybrid approach consisting of Linear Regression to leverage its efficiency and accuracy, and Support Vector Machine to handle malformed data in higher dimensions, where pattern fitting is easier. The features we will use are mood and price change over a three-day window, confidence of tweet sentiment and price change over a three-day window and volume of sentiment strength and price change over a three-day window. A three-day window was selected, because Chen and Lazer showed that best results arise when Twitter data predates the market movement by about three days [4].

Stock Prediction Model

In this section, we illustrate the method used to train a series of classifiers for predicting the polarity of the daily market opening price change for two stocks namely Apple (APPL) and Google (GOOG). As mentioned above, these were chosen primarily because they are all actively discussed in social media, specifically Twitter. Because we downloaded our training data through the Twitter Firehose API, we only had access to 1% of the Twitter corpus. Thus, we expect the approach defined here to be applicable to a larger cohort of companies and on a larger scale as we begin to access more of the Twitter graph. The figure below shows an overview of the stock prediction model.

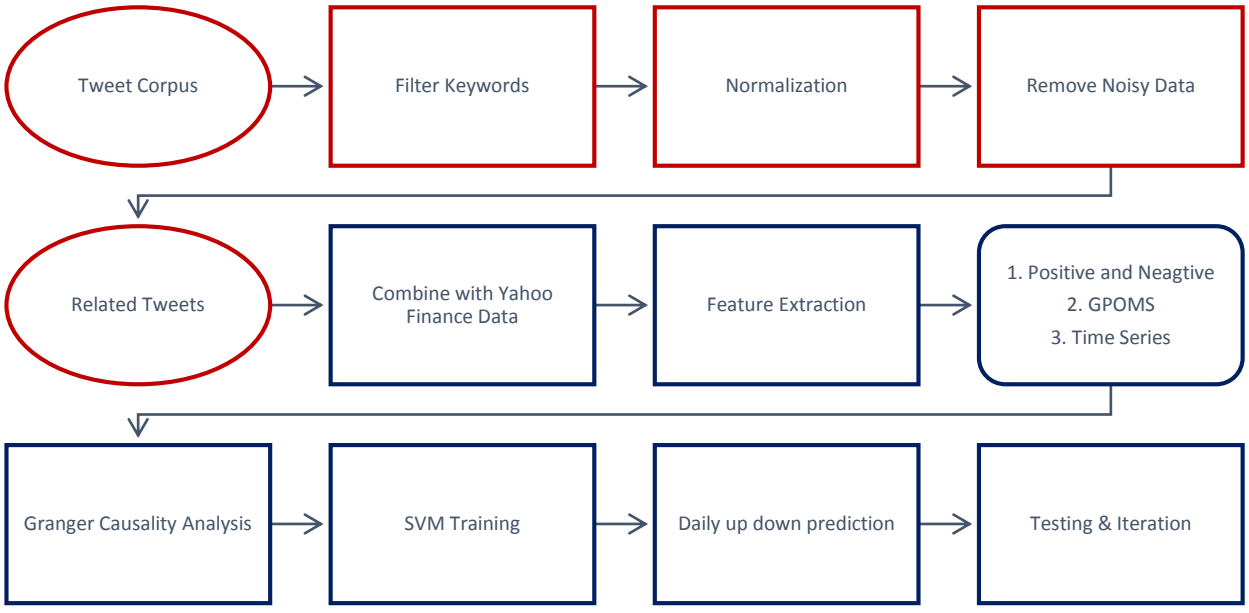

Figure 2 Overview of Stock Prediction Model

Data Preprocessing

Shaping the raw data into a malleable training set is an important part of the model. This stage contains all the steps from ingesting the Twitter data to analyzing this data with natural language processing techniques in order to provide structured and meaningful data for the machine learning process.

Ingestion and Normalization

For the algorithm, we ingested tweets using the Twitter API from October 7th, 2013 to November 30th 2013 solely from North America. This geographic sampling is significant as it benefits in harmonizing vocabulary. It allowed us to obtain a set of tweets highly concentrated from a consistent English vocabulary. Over the course of this period, we collected approximately 120,000,000 (120 million) tweets. It is important to note that we are still continuing to obtain data for further iterations and refinement of our algorithm. Before processing, we normalized the original Twitter data time stamps to American Eastern Time as to provide consistency with the NASDAQ stock market time.

Data Selection and Noise Filtering

From this corpus of tweets, we selected relevant data using Google Adwords to create a list of terms for each stock. We filtered the set of ingested Tweets according to this list of keywords. Below is an example of this keyword list.

Google: Goog, Google, Android, Glass, KitKat, Chrome, AdWords, AdSense, gmail, youtube etc.
Apple: Appl, Apple, IOS, iphone, mac, ipad, siri, apple store, itunes, macbook etc.

After keyword selection, we query for related tweets.

Concerning tweets, it has been well established (Han and Baldwin, 2011) there is an issue of lexical normalization for out-of-vocabulary (OOV). These words include colloquialisms, slang and words spelled differently to emphasize emotion such as "cooooooooooolll, LOL." Since around 15% of Tweets have more than 50% OOV tokens (Han and Baldwin, 2011), we used the method proposed by Han to normalize these tweets and filter out noise.
Furthermore, a proportion of tweets mentioning keywords often times have no relation to the stock company at all. Instances like "I need some mac & cheese" are identified as related tweets although they are noise. We use a Name Entity Recognition (NER) system to identify entities in tweets and then remove unrelated entities. We used the open source Stanford Named Entity Recognizer based on linear

conditional random fields (CRF). If a Tweet doesn't contain any named entities as listed on the company keyword list, it is removed.

Machine Learning

The result of data preprocessing is a set of relevant tweets distributed between Google and Apple. With this data, we then combine it with financial data, apply natural language processing techniques (NLP) to extract features, and then train on it to arrive at a classifier.

Finance Data Amalgamation

We divided the tweets into three day windows per *Bollen and Al* as they identified it as an optimal window. We then classified the stock market data into two classes: raise, drop.

$$Change_i = OpenP_i - CloseP_{i-1} = \begin{cases} difference \geq 0 & ,Raise \\ difference < 0 & ,Drop \end{cases}$$

Where *i* is the stock market open day, $Change_i$ is stock market change on day i. We look at the opening price of that day and compare it with the closing price of the previous day. This change is greater than 0 we classify that day as raise and vice-versa. Since the constraining factor was the range of tweet collection, we only had 45 days of stock market open days excluding weekends and public holidays. We then divided up the tweets into segments of three days per company and associated a value with this three day window. This value was the $Change_i$ where *i* is the day following this three day window. By doing this, we obtained a two dimensional vector:

$$\{T(i-3, i-1), Yi\}$$
$$T_{uv} = T(u, v) = All\ tweets\ between\ day\ u, and\ day\ v, inclusive$$
$$Yi = Change_i\ for\ day\ i$$

Feature Extraction

We extracted 6 features using natural language processing techniques for each T_{uv}.

I. Average polarization: Binary feature of whether the average of all tweets in T_{uv} was positive or negative.

$$AvgPol_i = \frac{1}{N}\sum_{n=0}^{N-1} Pol(T_n) \;\; n \in N, where\ N = total\ number\ of\ Tweets\ in\ T_{uv}$$

II. Current polarization: Average polarization of the tweets in a 1 day window predating *i*

$$CurPol_i = \frac{1}{N}\sum_{n=0}^{N-1} Pol(T_n) \;\; n \in N, where\ N = total\ number\ of\ Tweets\ in\ T_{i-1}$$

III. Polarization Time Series: 3 day window of average polarization per hour predating *i*

$$SerPol_i = [Pol(T_1), Pol(T_2) \dots Pol(T_n) \dots Pol(T_N)], \quad SerPol_i \in R^N$$
$$n \in N, where\ N = total\ number\ of\ Tweets\ in\ T_{uv}$$

IV. Average sentiment: 6 dimensional vector for the average tweet sentiment in T_{uv}

$$AvgSent_i = \frac{1}{N}\sum_{n=0}^{N-1} Sent(T_n) \;\; n \in N, where\ N = total\ number\ of\ Tweets\ in\ T_{uv}$$
$$Sent = [x_1, x_2, x_3, x_4, x_5, x_6]$$

Where Sent is a multidimensional vector representing the coefficients of the GPOMS results (calm, alert, sure, vital, kind, happy).

V. Current sentiment: 6 dimensional vector for the average tweet sentiment in a 1 day window predating *i*

$$CurSent_i = \frac{1}{N}\sum_{n=0}^{N-1} Sent(T_n) \;\; n \in N, where\ N = total\ number\ of\ Tweets\ in\ T_{i-1}$$

VI. Sentiment Time Series: 3 day window of average polarization per hour predating *i*

$$SerSent_i = [Sent(T_1), Sent(T_2) \dots Sent(T_n) \dots Sent(T_N)], \quad Sent_i \in R^N$$
$$n \in N, where\ N = total\ number\ of\ Tweets\ in\ T_{uv}$$

These features formed a series of vectors for each day we analyzed:

$$F_i = [AvgPol_i, CurPol_i, SerPol_i, AvgSent_i, CurSent_i, SerSent_i]$$

When combined with the finance data, we were able to obtain a binary data set, the necessary form for applying support vector machine based supervised learning.

$$D_i = [F_i, Y_i]$$
$$Y_i = Change_i$$

Granger Causality Analysis

Although an indicator or variable X may correlate with another variable Y, it is incorrect to say that it is a causation. If Y is said to be caused by X, then X is then a strong predictor for Y. Although by simple inspection, we were able to ascertain that our features correlated with the change in stock, in order for them to effectively predict the change, there should be an indication of causality. A sufficient enough test that determines not quite causality but "Granger Causality" which implies that X has predictive information about Y is the Granger Causality test.

In the Granger Causality test, our null hypothesis is that a time series of our features do not predict fluctuations in the stock markets of GOOG and APPL. By lagging our feature time-series behind the change in stock market, and examining the significance values outputted by the Granger Causality test, we are able to reject the null hypothesis as P was less than 0.05 ($P < 0.05$). This definitively illustrated a statistical predictive relationship between our features and the movement of stock.

Support Vector Machine

Support vector machine is a machine learning technique used to classify binary data by creating a hyperplane with maximum margins separating the data set. If the data is not linearly separable, then we can transform the data into higher dimensions similar to holographic duality (where all 3 dimensional data is contained in 2 dimensions) using some predefined transformation function and then attempt to establish another optimum hyperplane. Given $D_i = [F_i, Y_i]$, we wish to optimize the following problem.

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i$$

$$s.t.$$

$$Y_i(w^T \emptyset(F_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

The training vectors F_i are mapped to a higher dimensional plane by the function $\emptyset$. SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. C > 0 is the penalty parameter of the error term. Furthermore, $K(F_i, F_j) = \emptyset(F_i)^T(F_j)$ is called the kernel function. The kernel function is used to define the structure of the hyperplane. The kernel function we used was the conventional radial basis function (RBF):

$$K(F_i, F_j) = exp\left(\gamma \left|\left|F_i - F_j\right|\right|^2\right), \gamma > 0$$

Where γ is a kernel parameter.

Testing & Iteration

We tested various permutations of features for F_i. Our experimental results corroborated our intuition which was that the permutation with all 6 features would produce the most accurate results.

Table 1 Feature Permutations

Features	Apple Accuracy	Google Accuracy
CurrPol + CurSen	51.63%	52.23%
AvgSent + AvgPol	57.22%	56.79%
AvgSent + CurrSent + CurPol + AvgPol	64.37%	63.46%
AvgSent + CurrSent + CurPol + AvgPol + SerPol + SerSent	73.17%	75.61%

These accuracy results were determined using simple 10-fold cross validation where subsets of our training data are used as testing data. Accuracy represents the percentage of the data in which our algorithm correctly predicted the change. As the feature set composed of all the features provided the best results, we used it in our final prototype algorithm. We used a 60/40 split of our training data to train our classifier. Depending on where our algorithm failed we would move the failure cases from our testing set into our training set to increase accuracy. This was an iterative process.

Experimental Results & Conclusion

Below is the output of our best classifier.

Table 2 Experimental Results

Stock	Accuracy	Training Set	Testing Set
GOOG	76.81%	~6000 tweets/day for 45 days	10 days
APPL	78.32%	~7000 tweets/day for 45 days	10 days

The state of art is 87.8% accuracy in prediction for the daily fluctuations of the Dow Jones Industrial Average by Bollen et Al in 2011. Albeit, their results are more accurate, they used an industrial average in contrast to the specificity we provide.

This research conclusively shows that we can predict the daily fluctuations of stock using twitter sentiment. The training set we used was limited to 45 days. With more iterations, a larger training set and more complex features, we are confident that we will be able to surpass the state of art.

System Design

Architecture Design

High-level design:

The high level architectural design consists of a web application that will be built on top of an existing web development framework, with a middle-tier to support the communication between the front-end and back-end services using a RESTful API. Each backend will store signals (features extracted from the tweets) in a database. The data will be processed through a machine-learning algorithm to compile the

transaction results for each company. Finally, the web application will read this data from the database and render it into a user interface.

Solution Criteria:

The criteria used to select the specific design consisted of both quantitative and qualitative measures. The quantitative measures referred to the performance measures attributed to large-scale web applications (running time, memory usage, page response time, server costs), while the qualitative measures assessed the user interaction and usability of the application.

Final Design, Critical Components and Pricing:

In order to satisfy the solution criteria defined above, the final design that was chosen and implemented combined the optimal individual selections for the algorithm choice, database, web framework, and deployment strategy. The database chosen for the design was PostgresSQL, as it was the most preferred database, for both performance reasons, as well as ease of use and extensibility. Furthermore, all group members had experience with Postgres, and felt comfortable moving forward with it as the underlying database. Heroku was the deployment infrastructure that was chosen, since all group members were comfortable with its technology, and it had a great performance to price ratio. Finally, the web application development framework chosen was Sinatra. It is a lightweight ruby-based rapid prototyping framework that provided out-of-the-box support for the database of choice, Postgres. Machine learning models of various types (Support Vector Machines, Artificial Neural Nets and Linear Regression models) will be used in tandem to determine optimal trade execution. The specifics regarding the machine learning algorithm that was chosen to be used will be described and analyzed in detail in the Algorithm Design section.

As described in the previous section, the three main components used for the design are the PostgressSQL database, a Heroku deployment infrastructure and the Sinatra framework. The pricing for this technology, for the purpose of this design project, was essentially negligible. While the pricing for Heroku is stated as $0.05/hr for each computational hour, Heroku also provides a free service that allow for instances of lightweight applications to be deployed for free. The other two components, Postgress and Sinatra, are both also available free of cost.

The block diagram below illustrates the overall architecture and technologies that was used:

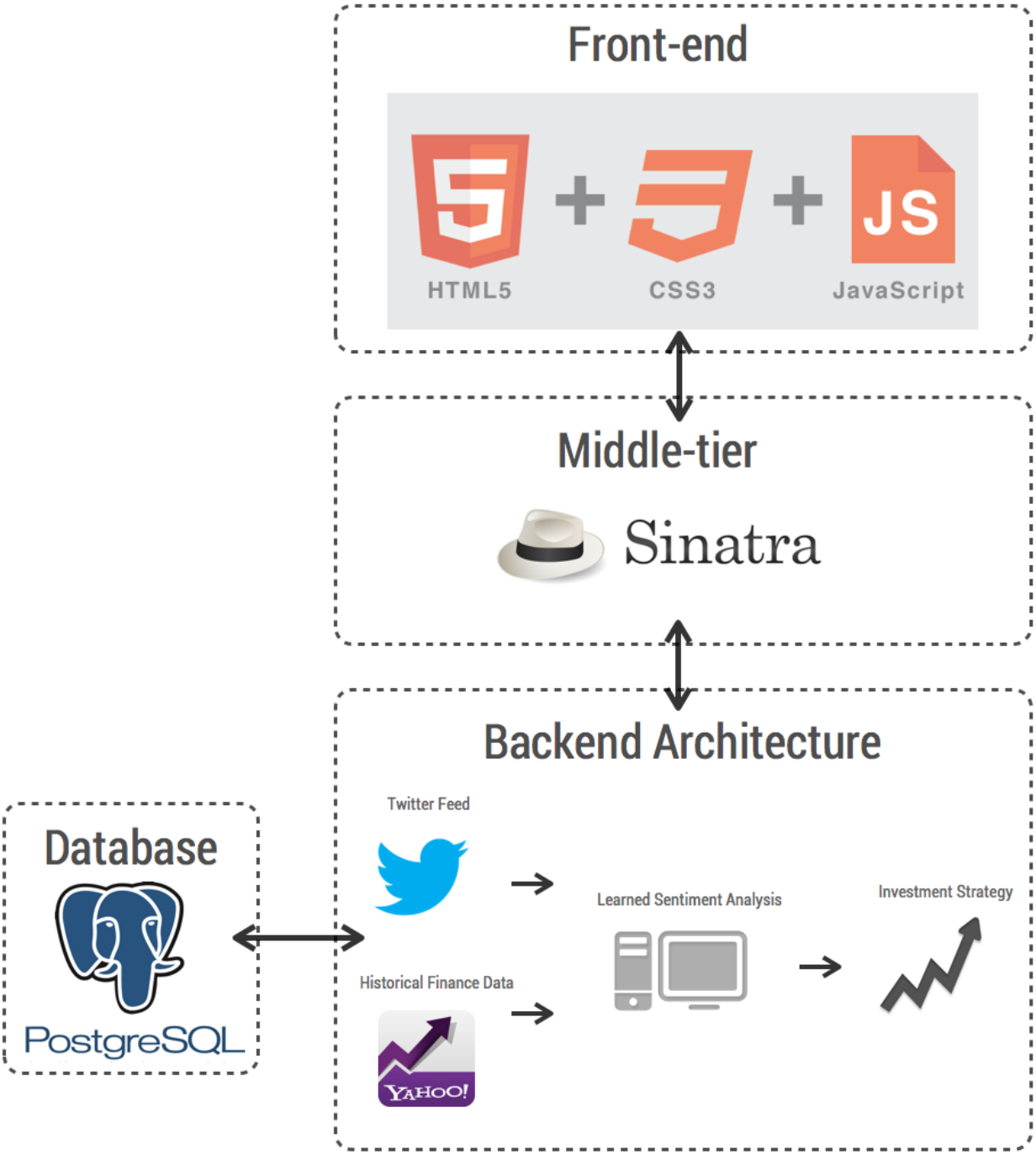

Figure 3 Proposed Architecture

User-Interface

The screenshots provided below shows the various webpages of the user-interface used for the low-fidelity prototype.

Figure 4 User Interface

This image above is the homepage for Socci. The user is allowed to select the company he or she is interested in viewing. For the purpose of the low-fidelity prototype, Apple and Google were the two options available to the user.

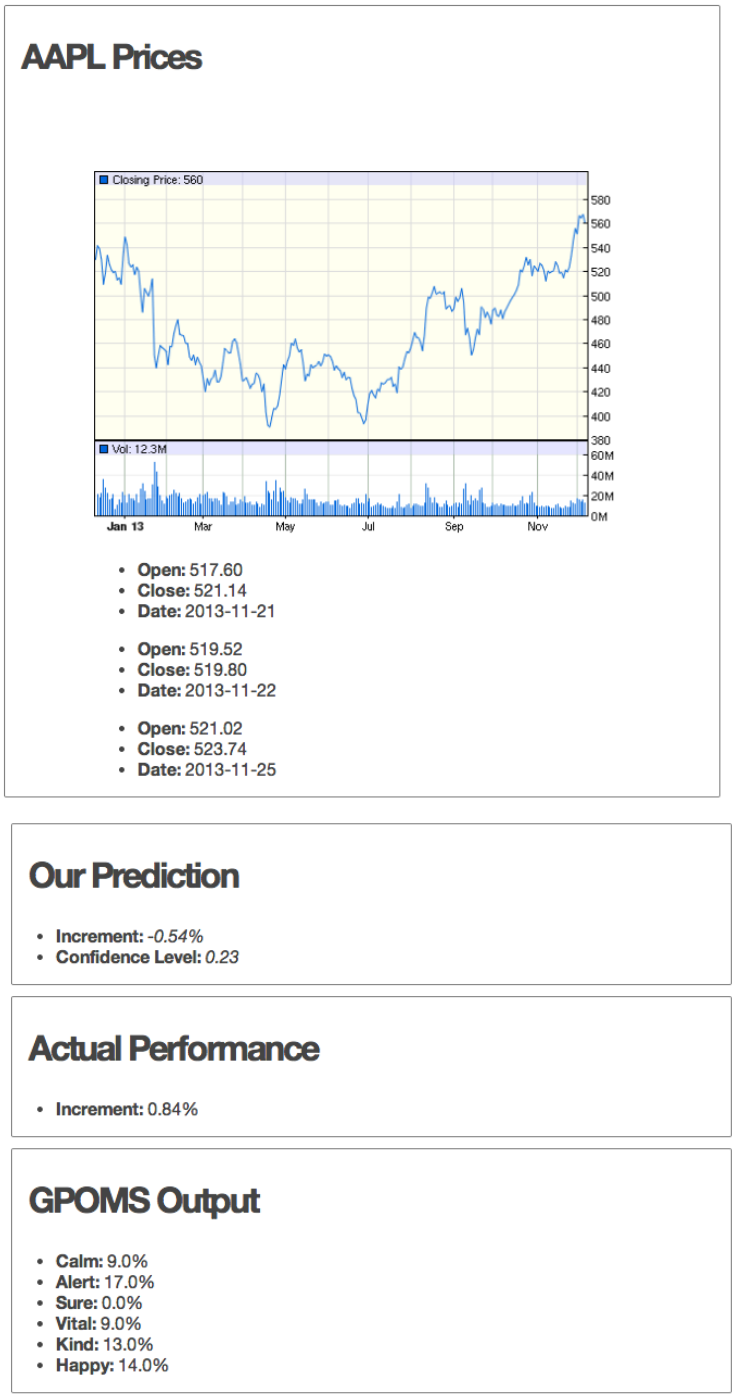

Figure 5 Prediction Interface

The above image is of the page that is linked to once a user clicks on “Apple”. For the purpose of the low fidelity prototype, the page provides the user with data computed through our backend machine learning algorithms. It provides an array on information; a graphical illustration of Apple’s stock trends throughout

time, a prediction along with a confidence level, the actual performance of the stock for the given day and the GPOM output computed using the backend algorithms.

Alternative Comparisons and Test Procedures:

During the design phase there were three main design approaches being considered.

1. An application that used a SQLite database on the backend, with Play as the web-application framework and EC2 as the deployment service.
2. An application with MongoDB as the database, Flask as the framework and Google Compute Engine to host the application.
3. An application that used PostgresSQL as the database, Sinatra as the web-application framework and Heroku as the hosting service. This was the eventual design choice.

Each of the specific components, i.e. the database, web framework and deployment infrastructure, were compared against one another, using the quantitative and qualitative measures in the solution criteria defined earlier.

The scalability of each of the databases were compared by analyzing the various features they provided and measuring their read/write performance through automated test.

SQLite is incredibly popular as an embedded database, and is, arguably, the most widespread database engine. It also provides bindings to many popular programming languages, which helps the development process. Unfortunately, very large, single tables do not perform well under SQLite and joins across multiple large tables are essentially impossible to complete in a timely manner.

Database

MongoDB is a document-oriented database; it is not a relational database, rather, it is classified as a NoSQL database. It favours JSON-like documents with dynamic (changing) schemas, making data integration easy and fast for web-applications. MongoDB is the most popular commercial NoSQL database system. MongoDB has been shown to be incredibly performant for web-based applications, however, it is now well understood by the development community, and has a significant cognitive overhead for development purposes.

Finally, PostgresSQL (often called Postgres) is an open sourced object-relational database management system. It is extremely extensible and standard compliant, and is licensed under the PostgresSQL License (open-sourced). It is fully-ACID compliant, transactional, and is known for being quite easy-to-use. Postgres also handles concurrency and parallelism through a process known as multi-version concurrency control, which gives each unique transaction a "snapshot in time" of the database, allowing it to independently make transactions until commit points. This helps it reduce the need for read locks, and ensures the database is ACID. Also, it allows multi-tiered atomicity, and is considered a highly consistent, available database. Postgres has been studied in many different circumstances, in regards to its performance. Recent versions of the database have shown incredible improvements in speed and scalability.

An automated test script was written to measure and compare the performance of the various databases. The script wrote/read one thousand, ten thousand and one hundred thousand rows from an instance of each database running on a local machine and tracked the amount of time (milliseconds) the task took for each database. The purpose of this test is to understand how database performance scales as the number of actions performed increases. The results from this test procedure are tabulated below.

Table 3 Read and Write times

Read Time (ms)				
	1,000 Rows	10,000 Rows	100,000 Rows	1,000,000 Rows
SQLite	0.02	0.03	0.08	0.21
MongoDB	0.02	0.05	0.10	0.27
PostgresSQL	0.01	0.01	0.02	0.08

Write Time (ms)				
	1,000 Rows	10,000 Rows	100,000 Rows	1,000,000 Rows
SQLite	0.09	0.14	0.35	3.90
MongoDB	0.04	0.4	1.5	2.90
PostgresSQL	0.01	0.04	0.10	0.87

It is evident from the results that the performance of PostgresSql scales the best as read/writes to the database increase. When this is factored in with the various beneficial features that Postgres provides, along with the drawbacks that come with the other database options, Postgres is the optimal choice.

Deployment

Deployment costs of the various application deployment services were compared against one another to determine the optimal choice. Ideally, for the scope and goals of this project, a low cost option

Amazon's EC2 is the virtualization component of Amazon's cloud-computing platform through which users can spin up virtual machine instances for computational purposes.
Amazon's EC2 starts at free-tier, where users are given 750 hours of a Unix or Windows machine. 750 hours of elastic load balancing, 30 gb of storage, and a 15 gb bandwidth across AWS services. The monetary cost for this service is negligible, since the amount of data processed significantly underutilized the services provided in this tier.

Google Compute Engine is an infrastructure-as-a-service product from Google, which uses KVM as a hypervisor allowing users to spin up Linux instances for computational purposes. It operates on top of a Debian based operating system and provides a REST-based API for end-user consumption. This service is free for educational purposes.

Heroku is a cloud platform that supports several programming languages and allows users to deploy their applications onto the real internet. The pricing for Heroku has $0.05/hr for each computational hour, and gives 512 MB of ram per instance. Heroku also provides a free hosting service for lightweight applications to run on, which is available for all users that sign up and create account.

After comparing the various deployment options, and considering the fact that the application will be very lightweight, Heroku's free services would be the most cost-efficient option.

Framework

The three frameworks, Play, Flask and Sinatra, were compared in order to choose the one that best fit the needs of this project.

The Play framework is a Java/Scala based web-application framework that defines specific classes and functionality for realizing the MVC architecture. The three main divisions in the MVC architecture are all mapped to classes in the Play framework. Play provides a wide variety of features, including hot-reload and a complex templating engine. While, it is very robust the learning curve is quite steep and requires an experienced developer to be build applications quickly.

Flask is a lightweight web application framework in Python, based on the Jinja2 templating engine. It is a micro framework, but is heavily extensible, and bares striking resemblance to the Sinatra web framework. A few drawbacks are that it does not provide native support for PostgresSql and that the team members do not have experience using it.

Sinatra is a Ruby-based rapid prototyping web framework. It provides "out-of-the-box" support for MVC, and allows users to include any functionality from Ruby on Rails (the full-fledged 13 Ruby web development framework). All group members have experience and familiarity with Sinatra.

After rigorously comparing the three frameworks' features, the framework chosen was Sinatra. This is because the developers on the team are most familiar with the technologies that come with Sinatra (i.e. Ruby on Rails and Python). Also, the out-of-the-box supporting features and ease of development with Sinatra fit best with the time constraints given for this project.

System Testing Procedures

The entire system, the software architecture in conjunction with the backend algorithm, was tested to ensure the application worked as intended. An integration test was performed, where the software built to support the application was integrated with the algorithm code base. A single well-known company, i.e. Apple, was used as the baseline for the integration test plan. The user-interface was built to support Apple as the clickable link. Once the link was clicked the user was directed to a second page with the data that was computed through the backend system (machine learning algorithm). This information was then cross-referenced with online financial trackers and the financial predictions made for the past three-days were also verified. This was performed a multitude of times to the system provided accurate results for

out Apple test plan. The next step was the addition of a second company (Google), which was tested in a similar manner to validate the scalability of the application.

Implications

A project of this nature inherently has large social and economic implications. First and foremost, however, this is a for-profit research based project. The goals of this study are to determine optimal techniques for profit maximization based on social signals. It is not primarily positioned to have positive societal or economic impact. However, if positive social and economic effects come about as a result of this project, they will not be ignored and will rather be acknowledged and improved as necessary.

Investments are primarily backed on real-world assets, for instance, mortgages backing houses or loans backing education. In order to have maximal social and economic positive impact, each company that is analyzed can be ranked based on their social contributions in the last year. Then, when the model predicts movement for each company, the top recommended companies can be displayed alongside their social contribution score. This will allow investors to be aware of the social implications of investing in a given company, essentially empowering them to make educated decisions.

Financial Analysis

Due to the nature of this project and the available online services (computer-based data analysis and research), there is no significant incurred cost.

Execution Plan

Team Structure

The organizational structure of the team is organized according to each person's strengths and the roles necessary for completing the projects Team strengths/weaknesses are found in Table 1 and our team roles can be found in Table 2.

Team Member	Strengths	Weaknesses
Vithu Jeya	Machine Learning, Financial Models, Product Strategy, Product Vision,	Application development
Harsh Dave	Product Design, Technical Writing, Front End Web Development	Back-end coding, mobile development
Ziyad Mir	Algorithm Design, Financial Models, Back End Development, Application Development	Front-end coding
Senduran Indrakumar	Front End Development, Back End Development, Application Design, Mobile Development	Product Design

Table 4 Team Weaknesses & Strengths

Given the small sizes of teams and a project with a fairly large scope and 5 major components, roles were allocated to members to find the best marriage between skillset and need. Some overlaps were needed based on roles that would require a larger contingent in order to meet our project objectives.

Table 5 Team Roles

Role/Team	Member(s)
Product Manager	Vithu Jeya
Researcher	Vithu Jeya, Ziyad Mir
UI/UX Design	Harsh Dave
Frontend development	Senduran Indrakumar, Harsh Dave, Ziyad Mir,
Backend development	Ziyad Mir, Senduran Indrakumar

Deliverables & Tasks

The main deliverables have been broken down into 5 components consisting of main subtasks. It is important to note that this early on into project planning, the tasks act as guidelines and are subject to change.

1. **Sentiment Analysis**
    a. Gather Twitter data
    b. Tag sentiment of data to create ground truth
    c. Write NLP Sentiment analysis algorithm
    d. Test Sentiment analysis
    e. Iterate

2. **Machine Learning Model**
    a. Collect sentiment analysis & trading data
    b. Tag ground truth
    c. Create classifiers for features
    d. Implement SVM + Linear Regression
    e. Train and Test model
    f. Iterate

3. **Back End**
    a. Create Postgres database
    b. Integrate Yahoo Finance & Twitter API's
    c. Integrate Machine Learning model
    d. Write Investment Strategy

4. **Front End**
    a. Set up Sinatra
    b. Design Front End
    c. Create Assets
    d. Implement Front End

5. **Course required work**
    a. Design Critique & Prototype Demo
    b. Design Brief & Outreach Presentation

Timeline & Gantt Chart

In our design proposal, we aimed to have our prototype built for November 25th 2013. This means, we wanted to have a prototype sentiment analysis algorithm, machine learning model and investment strategy ready to for simulation in order to verify the quality of our project. We met these goals to varying degrees of success. The full application, including the front and back ends will be built in line with a broader 8 month timeline with the final project target completion date being: April 1st, 2014. The Gantt chart in Figure 1 is a rough estimate for the timeline of the project. It shows our revised progress over the course of the past 4 months and goals for the final project. The tasks have been colour-coded for each role. The chart is colour coded according to which team will perform which task. However the course work milestones will be completed by all our group members.

Red: Product Managers

Dark Blue: Back End Engineers

Yellow: Researchers

Light Blue: Front End Engineers

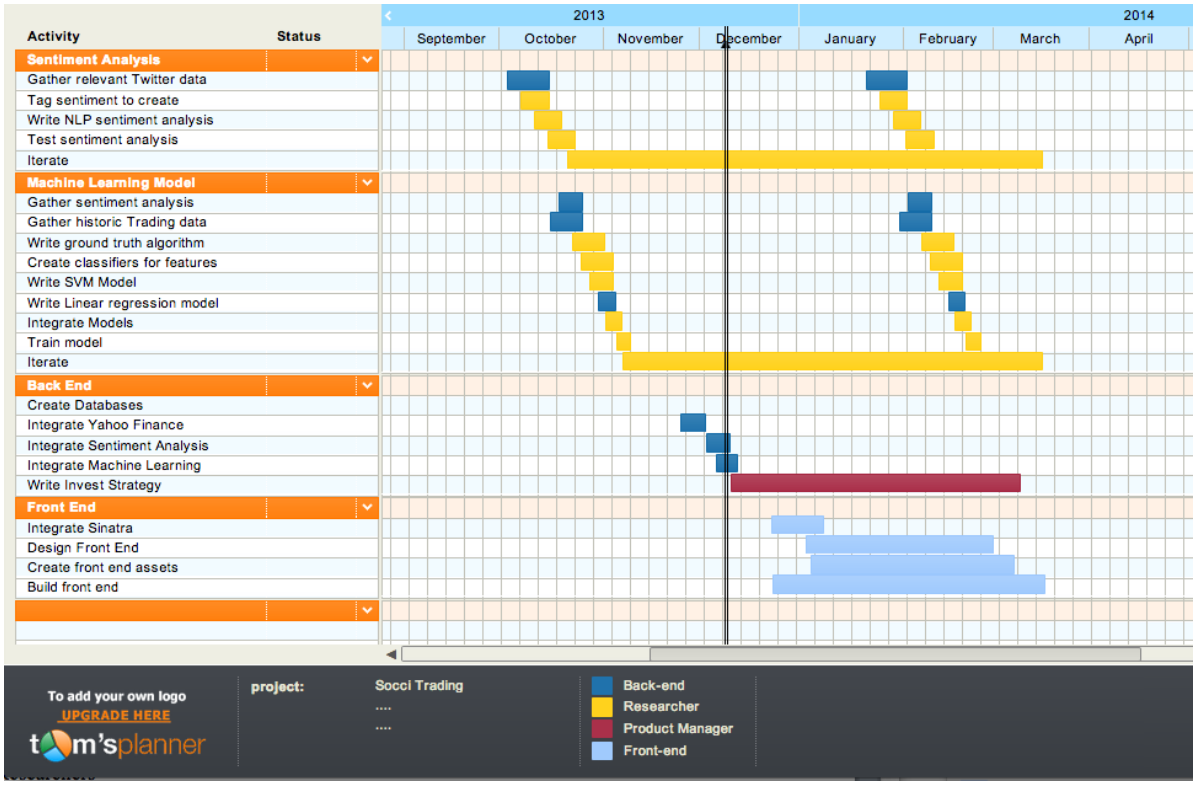

Figure 6Task Gantt chart

Winter Term Milestones & Tasks

The main goal for the duration of this fall term is to have interface built, and to increase our accuracy. Each role will have a significant part to play for the duration of the semester. The Front-end will not be as sophisticated as it will be mostly done in April, but the goal is to have most of the functionality surrounding it in place.

Machine learning Model: During this winter semester, we aim to test out our proposed algorithm by continuously iterate on it based on simulations.

Front End and Backend: We aim to begin formalizing our front-end and giving it a more sophisticated feel.

Conclusion

Historically, stock market movements have been highly unpredictable. With the advent of technological advances over the past two decades, financial institutions and researchers have developed computerized mathematical models to maximize their returns while minimizing their risk. One recent model by Johan Bollen involves analyzing the public's emotional states, represented by Twitter feeds, in order to predict the market [3]. However, after examining the state of art, we decided to attempt our own approach and develop a more robust, and accurate trading model based off of machine learned Twitter sentiment. We were able to demonstrate a 76% and 78% accuracy in predicting the daily fluctuations of two stocks: Google and Apple respectively. Our approach involved querying relevant tweets for a window of 3-day predating the prediction date, analyzing using the GPOMS natural language processing, then amalgamating this with stock data to ultimately train a classifier using support vector machines.

Improving and utilizing algorithms to better judge and anticipate stock market trends has become a staple of the finance industry. Meanwhile the global community is more connected than ever with the use of social networks such as Twitter, where people voluntarily post their thoughts and preferences. Today, most stock market predictions are made using historical data and mathematical modelling. As machine learning and NLP techniques becomes more prominent in its ability to gauge sentiment, it potentially gives us another tool to predict stock market performance. Socci Trading aims to leverage an untapped market in the field of stock market trading by using NLP and machine learning algorithms to judge the sentiment of public tweets, and provide predictions for how the stock market will move. The end result will be an application that will provide day by day stock fluctuation predictions based on a Twitter data from a 3 day window. We will then use this information and apply it to our investment strategy in order to increase the total value of our assets. The project will extend from September to April. Over the last four months, we've developed a rudimentary machine learning model which we will continue to iterate on. We are confident based on our weekly results that within the next four months we will be able to compete with the state of art. In the next four months, we will continue to improve our algorithm using the delineated process as well as build a polished interface. Furthermore, we plan on diversifying our portfolio and potentially outline a viable business model. Ultimately, we hope that our application and research can be used as a new foundation for further research in sentiment based trading methods in order to increase profits.

References

[1] Statistic Brain, "Facebook Statistics," Statistic Brain, 05 10 2013. [Online]. Available: http://www.statisticbrain.com/facebook-statistics/. [Accessed 05 10 2013].

[2] Statistic Brain, "Twitter Statistics," Statistic Brain, 05 10 2013. [Online]. Available: http://www.statisticbrain.com/Twitter-statistics/. [Accessed 05 10 2013].

[3] H. M. X. Z. Johan Bollen, "Twitter mood predicts the stock market," Journal of Computer Science, p. 4, 1 August 2011.

[4] M. L. Ray Chen, "Sentiment Analysis of Twitter Feeds for the," 2011. [Online]. Available: http://cs229.stanford.edu/proj2011/ChenLazer-SentimentAnalysisOfTwitterFeedsForThePredictionOfStockMarketMovement.pdf. [Accessed 10 Oct 2013].

[5] M. Bryant, "Twitter sentiment and the stock market: SNTMNT's API helps traders pick the best time to buy or sell," The Next Web, 5 July 2012. [Online]. Available: http://thenextweb.com/apps/2012/07/05/Twitter-sentiment-and-the-stock-market-sntmnts-api-helps-traders-pick-the-best-time-to-buy-or-sell/. [Accessed 10 10 2013].

[6] Gallagher, L. A & Taylor, M. P. (2002) Southern Economic Journal 69,345–362.

[7] Kavussanos, M & Dockery, E. (2001) Applied Financial Economics 11,573–79.

[8] Butler, K. C & Malaikah, S. J. (1992) Journal of Banking and Finance16, 197–210

[9] Schumaker, R. P & Chen, H. (2009) ACM Trans. Inf. Syst. 27, 12:1–12:19.

[10] Gilbert, E & Karahalios, K. (2010) Widespread worry and the stockmarket.

[11] Gruhl, D, Guha, R, Kumar, R, Novak, J, & Tomkins, A. (2005) The predictive power of online chatter. (ACM, New York, NY, USA), pp.78–87.

[12] Mishne, G & Glance, N. (2006) Predicting Movie Sales from Blogger Sentiment. AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs

[13] S. Asur and B. A. Huberman 2010 Predicting the Future with Social Media arXiv:1003.5699v1

[14] Choi, H & Varian, H. (2009) Predicting the present with google trends., (Google), Technical report.

[15] Liu, Y, Huang, X, An, A, & Yu, X. (2007) ARSA: a sentiment-aware model for predicting sales performance using blogs. (ACM, New York, NY, USA), pp. 607–614.

[16] Dolan, R. J. (2002) Science 298, 1191–1194.

[17] Dodds, Peter. (2009) Journal of Happiness July, doi: 10.1007/s10902- 009-9150-9

[18] Damasio, A. R. (1994) Descartes' error : emotion, reason, and the human brain. (Putnam), pp. xix, 312 p.+.

[19] Nofsinger, J. (2005) Journal of Behaviour Finance. 6, 144–160.

[20] Edmans, A, Garca, D, & Norli, . (2007) Journal of Finance 62, 1967– 1998.

[21] Hirshleifer, D & Shumway, T. (2003) Journal of Finance 58, 1009–1032.

[22] Pak, A & Paroubek, P. (2010) Twitter as a Corpus for Sentiment Analysis and Opinion Mining. (European Language Resources Association (ELRA), Valletta, Malta).

[23] Pang, B & Lee, L. (2008) Foundations and Trends in Information Retrieval 2, 1–135.

[24] Wilson, T, Wiebe, J, & Hoffmann, P. (2005) Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. (Vancouver, CA).

[25] Wilson, T, Hoffmann, P, Somasundaran, S, Kessler, J, Wiebe, J, Choi, Y, Cardie, C, Riloff, E, & Patwardhan, S. (2005) OpinionFinder: A system for subjectivity analysis. pp. 34–35.

[26] O'Connor, B, Balasubramanyan, R, Routledge, B. R, & Smith, N. A. (2010) From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series.

[27] Diener E, Diener M & Diener D (2009) Factors Predicting the Subjective Well-Being of Nations. Social Indicators Research Series 38:43-70

[28] Lapedes, A & Farber, R. (1987) Nonlinear signal processing using neural network: Prediction and system modeling, (Los Alamos National Lab Technical Report), Technical report.

[29] Zhu, X, Wang, H, Xu, L, & Li, H. (2008) Expert Syst. Appl. 34, 3043– 3054.