Stock Price Forecasting via Sentiment Analysis on Twitter

John Kordonis
Electrical and Computer Engineering Dept.
Democritus University of Thrace
Xanthi 67 100, Greece
ioankord1@ee.duth.gr

Symeon Symeonidis
Electrical and Computer Engineering Dept.
Democritus University of Thrace
Xanthi 67 100, Greece
ssymeoni@ee.duth.gr

Avi Arampatzis
Electrical and Computer Engineering Dept.
Democritus University of Thrace
Xanthi 67 100, Greece
avi@ee.duth.gr

ABSTRACT

Stock price forecasting is an important and thriving topic in financial engineering especially since new techniques and approaches on this matter are gaining ground constantly. In the contemporary era, the ceaseless use of social media has reached unprecedented levels, which has led to the belief that the expressed public sentiment could be correlated with the behavior of stock prices. The idea is to recognize patterns which confirm this correlation and use them to predict the future behavior of the various stock prices. With no doubt, though uninteresting individually, tweets can provide a satisfactory reflection of public sentiment when taken in aggregate. In this paper, we develop a system which collects past tweets, processes them further, and examines the effectiveness of various machine learning techniques such as Naive Bayes Bernoulli classification and Support Vector Machine (SVM), for providing a positive or negative sentiment on the tweet corpus. Subsequently, we employ the same machine learning algorithms to analyze how tweets correlate with stock market price behavior. Finally, we examine our prediction's error by comparing our algorithm's outcome with next day's actual close price. Overall, the ultimate goal of this project is to forecast how the market will behave in the future via sentiment analysis on a set of tweets over the past few days, as well as to examine if the theory of contrarian investing is applicable. The final results seem to be promising as we found correlation between sentiment of tweets and stock prices.

Keywords

Stock Market Prediction, Sentiment Analysis, Twitter, Machine Learning, NLP

1. INTRODUCTION

Modern data mining techniques have led to the development of sentiment analysis, an algorithmic approach for detecting the predominant sentiment about a product or company using social media data. A prominant field for the use of sentiment analysis has been stock market forecasting, a subject undeniably undergoing intense studies [2][3].

Nowadays, a great volume of data, which contains information about numerous topics, is being transmitted online through various social media. An excellent example is Twitter, where over 400 million tweets are sent daily. Though each tweet may not be significant as a unit, a large collection of them can provide data with valuable insight about the common opinion on a particular subject. Gauging the public's sentiment by retrieving online information from Twitter, can be valuable in forming trading strategies. The correct prediction about the fluctuation of stock prices depends on many factors, and public sentiment is arguably included.

The rest of the paper is organized as follows. In Section 2 we describe related works about sentiment analysis on stock prediction and in following section we propose our model. In Section 4 we briefly discuss the datasets that we have used for this paper and data preprocessing methods adopted. Section 5 discusses the sentiment analysis technique we developed for the purpose of this work. Section 6 includes in detail, the different machine learning techniques to predict the sentiment scores. In Section 7, we use the predicted sentiment scores and investigate a technique for forecasting stock price by exploring the correlation between tweets and stock market. In Section 8, we summarize our results and propose ideas for further future work.

2. RELATED WORK

Alongside the rising number of blogs and social media, opinion mining and sentiment analysis became very popular and began to spread. An extensive synopsis of the existing work was presented in [7]. Wysoki's research project had as a goal to study stock market message boards for over 3,000 stocks to determine if a correlation between discussion board message volume and message quality had any effect on the volume or price for a stock. A key contribution of this research showed a strong positive correlation between the volume of messages posted on the discussion boards during the hours that the stock market was not open and the next trading day's volume and stock returns. The researchers stated that a tenfold increase in message postings in the overnight hours led to an average increase in the next day's stock volume of approximately 15.6 percent and a 0.7 percent increase in next day stock returns [7].

Along the same lines, [1] took into account messages posted on stock-related message boards and calculated their ability to affect the movements of stock prices. The researchers examined around 1.5 million messages posted on two message

PCI '16, November 10-12, 2016, Patras, Greece

DOI: http://dx.doi.org/10.1145/3003733.3003787

boards for 45 companies and performed text classification and sentiment analysis to define the sentiment of each message. With their work they were able to show a positive correlation between message board posts and stock price fluctuations on the day after [1].

While both of the above projects involved stock message discussion boards, attempts have been made in order for modern media to be included. In particular, many have used Facebook, Twitter and other social media platforms in order to predict the movement of stock prices or of the market in general. For example, [5] proposed the use of Twitter specifically as a corpus for sentiment analysis and refered to the methods of tweet gathering and processing. The researchers used specific emoticons to form a training set for sentiment classification, a method that curtailed the need to tag tweets by hand. Their training set was divided into positive and negative samples based on happy and sad emoticons [5].

Undeniably, the work of [2] is amongst the most popular on the field. In their publication, the researchers described the use of sentiment analysis of a large corpus of Twitter messages in order to understand the average mood of Twitter users on a spesific day. Afterwards, these results became the input into a neural network prediction engine in order to forecast the movement of stock prices on the very next day, with a reported 87.6 percent accuracy of prediction of the Dow Jones Industrial Average. Even though this project did indeed show a correlation between sentiment scores and stock price movements, the authors also proceeded in examining a vast amount of tweets derived from all Twitter users and not just from users directed toward the stock market. Their effort was made in order to grasp the overall sentiment of the Twitter population.

Furthermore, the paper of [3] achieves a 75 percent accurate prediction mechanism using Self Organizing Fuzzy Neural Networks on Twitter and DJIA feeds. In their research, they created a custom questionnaire with words to analyze tweets for their sentiment.

Last but not least, another important publication is that of [6]. They decided to follow a more straightforward path by focusing on the "Standard and Poor's" (S&P) top 100 stocks, collecting only related tweets. These were then analyzed in order to determine if the sentiment of a company on Twitter has any correlation to the movement in price or volume. [6] decided to follow the dollar symbol preceding the stock market symbol, nomenclature popularized by the stocktwitstook.com website and its users, in order to reduce the large amount of "noise" on Twitter. This nomenclature allowed them to collect only tweets that had been created and shared by users with an interest in the stock market. Their results indicate that the sentiment of a company on Twitter closely follows market movements and that message volume on Twitter is positively correlated to the trading volume for that stock [6].

The techniques proposed by these papers provide an interesting overview of sentiment analysis and how it can relate to the stock market. However, the results seem varied and may depend on the accuracy of a twitter sentiment classifier, as well as the preprocessing and filtering of tweets, including the process of choosing which tweets to use for analysis. In this work, we emulate some of these previous works to build a sentiment analyzer, but specifically for the Twitter domain.

3. OUR MODEL

In this paper, we mined tweets using Twitter's Search API and subsequently processed them for further analysis, which included Natural Language Processing (NLP) and Sentiment Analysis. Thereafter, we applied Naive Bayes and SVM to predict each tweet's sentiment. By evaluating each model for its proper sentiment classification, we discovered that Support Vector Machines give higher accuracy through cross validation. Despite this fact, we continued to take into consideration both techniques and compare every time their accuracy. After predicting every tweet's sentiment, we mined historical stock data using Yahoo finance API. We then created a respective feature matrix for stock market prediction using sentiment score and stock price's change for each day and at the end we proposed our own trading strategy.

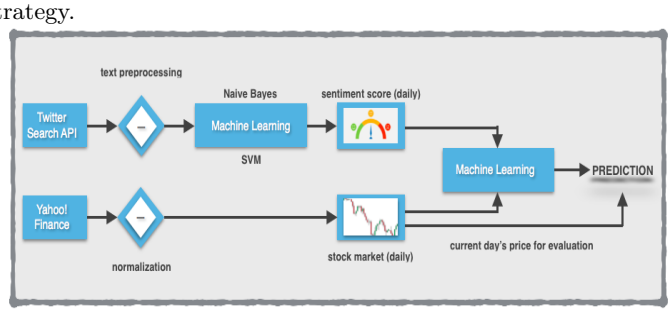

Figure 1: Flow diagram of Algorithmic model.

4. DATASET

We used two live fed datasets:

- Stock Prices obtained using Yahoo! Finance API. This dataset consists of the Open, Close, High and Low values for each day.
- We obtained also a collection of tweets using Twitter's Search API. For each tweet these records provide a tweet id, the timestamp and tweet text, which is by design limited to 140 characters and needs to be filtered from noise. Since we perform our prediction and analysis on a daily basis, we split the tweets by days using the timestamp as the main index of the dataframe.

Stock Data Preprocessing

The data we collected from Yahoo! Finance had to be preprocessed in order to become suitable for further reliable analysis. The main problem we faced was that while the Twitter data were available for each day of the week, including weekends, the stock values were absent for weekends and other holidays when the market is closed. In order to fill the missing values, we overcame this problem by using a simple statistical function. If the value that is missing is y, the previous known value is xprevious and the next known value is xnext, then the value y will be:

$$y = (\text{xprevious} + \text{xnext})/2.$$

This approximation works most of the time very well except in cases of unexpected rapidly rise and fall.

Furthermore, we create two additional fundamental metrics:

$$HLPCT = \frac{High - Low}{Low}$$

$$PCTchange = \frac{Close - Open}{Open}$$

HLPCT stands for "High-Low Percentage" and PCTchange for "Percentage change". Both metrics are important for the machine learning algorithm which we applied in order to find the correlation between tweets and stock market.

Twitter Data Preprocessing

For the process of collecting tweets, Twitter provides two possible ways to gather Tweets: the Streaming API or the Search API. The Streaming API allows users to obtain real-time access to tweets from an input query. The user first requests a connection to a stream of tweets from the server. Then, the server opens a streaming connection and tweets are streamed in as they occur, to the user. However, there are a few limitations of the Streaming API. First, language is not specificable, resulting in a stream that contains Tweets of all languages, including a few non-Latin based alphabets, that complicates further analysis.

Because of these issues, we decided to go with the Twitter Search API instead. The Search API is a REST API which allows users to request specific queries of recent tweets. The Search API allows filtering based on language, region, geolocation and time. There is a rate limit associated with the query, but we handle it in the code.

The request returns a list of JSON objects that contain the tweets and their metadata. This includes a variety of information, including username, time, location, retweets, and more. For our purposes, we mainly focus on the time and tweet text. We filter out the unnecessary metadata and store both the tweet text and its timestamp in a .txt file. We use as query the ticker of the company in front of which we add the dollar sign to gather the most "financial" tweets.

Both of these APIs require the user have an API key for authentication. Once authenticated, we were able to easily access the API through a python library called "Tweepy", which is a wrapper for the Twitter API.

5. SENTIMENT ANALYSIS

Text Processing

The text of each tweet includes a lot of words that are irrelevant to its sentiment. For example, some tweets contain URLS, tags to other users, or symbols that have no meaning. In order to better determine a tweet's sentiment score, before anything else we had to exclude the "noise" that occurred because of these words. For this to happen, we relied on a variety of techniques using the Natural Language ToolKit (NLTK) for Python.

A. Tokenization

Firstly, we divided the text by spaces, thus forming a list of individual words per tweet. We then used each word in the tweet as features to train our classifier.

B. Removing Stopwords

Next, we removed stopwords from the list of words. Python's Natural Language ToolKit library contains a stopword dictionary, which is a list of words that have neutral meaning and are inappropriate for sentiment analysis. To remove the stopwords from each text, we simply checked each word in the list of words against the dictionary. If a word was in the list, we excluded it from the tweet's text. The list of stopwords contains articles, some prepositions, and other words that add no sentiment value (able, also, or, etc.)

C. Twitter Symbols

It is not uncommon that tweets may contain extra symbols such as "@" or "#" as well as URLs. On Twitter, the word following an "@" (mentions) symbol is always a username, which we also exclude because it adds no value at all to the text. Words following "#" (hashtags) are not filtered out because they may contain crucial information about the tweet's sentiment. They are also particularly useful for categorization since Twitter creates new databases that are collections of similar tweets, by using hashtags. URLs are filtered out entirely, as they add no sentiment meaning to the text and could also be spams.

Training Set Collection

To train a sentiment analyzer and obtain data, we were in need of a system that could gather tweets. Therefore, we first collected a large amount of tweets that would serve as training data for our sentiment analyzer.

In the beginning, we considered manually tagging tweets with a "positive" or "negative" label. Thus we created a list of 1000 hand-classified tweets but because it was hard and time-consuming, we decided to look for a database with already sentiment-classified tweets. Surprisingly, we found that our search for other tweet corpuses returned no results, as Twitter had recently modified its terms of service to disallow public hosting of old tweets. Under these circumstances, we turned to alternative methods in order to form a training set. Specifically, we had two main ideas on how to classify tweets as training data.

According to the first idea we created a "positive" and a "negative" dataset for training, by using Twitter's Search API. Each dataset was created programmatically and was based on positive and negative queries on emoticons and keywords:

- Positive sentiment query: ":) :-) =) :D <3 like love"
- Negative sentiment query: ":(=(hate dislike"

Any tweet that included one or more of these keywords or emoticons was most likely to be of that corresponding sentiment. This resulted in a training set of "positive" and "negative" tweets which was almost as good as tagging tweets by hand.

The second idea was that we could maybe utilize a sentiment lexicon in order to classify the gathered tweets. The one we selected was the external lexicon AFINN[4], which is a list of English words rated for valence with an integer between minus five and plus five.

Training the classifiers

Once we had collected a large tweet corpus as training data, we were able to construct and train a classifier. Within this project we used two types of classifiers: Naive Bayes Bernoulli and Support Vector Machine. We chose to focus on these algorithms because according to [5], they are the state of the art for Sentiment Analysis. For both classifiers, we extracted the same features from the tweets to classify on.

A. Feature Extraction

A unigram is simply a N-gram of size one. For each unique tokenized word in a tweet, a unigram feature is created for the classifier. For example, if a negative tweet contains the word "bad", a feature for classification would be whether or not a tweet contains the word "bad". Since the feature came from a negative tweet, the classifier would be more likely to classify other tweets containing the word "bad" as negative.

Likewise, a bigram is a N-gram of size two and a trigram is a N-gram of size three. That means that in the case of bigrams the feature vector for the classifier is made of a two-word combinations and in the case of trigrams is made of a three-word combinations respectively. For example, if a negative tweet contains the combination "not perfect", in the case of the bigram feature extraction it would be classified as a negative tweet. Instead, if only unigram features were used, the tweet would have been classified as positive since the term "not" has a neutral sentiment and the term "perfect" a positive one.

B. Feature Filtering

With the method described above, the feature set grows larger and larger as the dataset increases leading to the point were it becomes difficult and unnecessary to use every single unigram, bigram, and trigram as a feature to train our classifier. So we decided to use only the n most significant features for training. We used a chi-squared test, Pearson's chi-squared test in particular, to score each unigram, bigram, and trigram in our training set. NLTK helped us to determine the frequency of each feature. Having, now, the features ordered by score, we selected the top-10000 to use for training and classification. This method undeniably speeded up our classifiers and reduced the amount of memory used.

Wordcloud

Having ordered the n most significant features we decided to create a Wordcloud in order to watch the significant features easily. Figure 2 shows an example of a Wordcloud for Apple's ticker "AAPL" on the 23rd of June 2016.

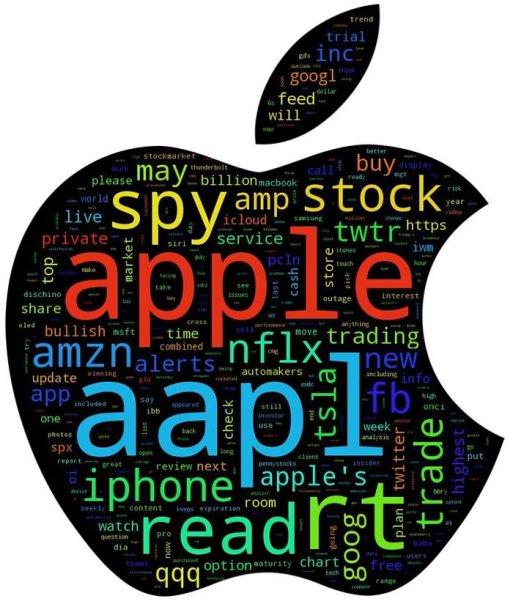

Figure 2: Wordcloud for stock "AAPL" (Apple).

6. MACHINE LEARNING TECHNIQUES

Accurate classification continues to be an engaging problem in machine learning and data mining. It is more than often that we need to create a classifier with a set of training data and labels. In our case, we want to build a classifier that is trained on our "positive" and "negative" labelled tweet corpus.This way, the classifier will be able to label future tweets as either "positive" or "negative", according to each tweet's characteristics and features. In this project, we examine two classifiers used for text classification: Naive Bayes Bernoulli and Support Vector Machines (SVM).

In the following examples, y represents the class label, which in our case is either "positive" or "negative", and x_i represents a feature in the feature set F.

Naive Bayes Bernoulli

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1,...,x_n|y)}{P(x_1,...,x_n}$$

Using the naive independence assumption that

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y)$$

for all i, this relationship is simplified to:

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

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y)$$
$$\Downarrow$$
$$\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class y in the training set. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

In our case, we use Naive Bayes' Bernoulli implementation which is distributed according to multivariate Bernoulli distributions. The decision rule for Bernoulli naive Bayes is based on

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

which differs from multinomial NB's rule in that it explicitly penalizes the non-occurrence of a feature i that is an indicator for class y.

The Naive Bayes Bernoulli classifier is extremely simple, and its conditional independence assumptions are not realistic in the real world. However, applications of Naive Bayes classifiers have performed well, better than initially imagined.

Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. Unlike the Naive Bayes classifier, the SVM is a large margin classifier, rather than probabilistic. In previous works, SVMs have been shown to be very effective for

text categorization. The SVM is a classifier that attempts to find a separation between a linearly separable set of data, with as wide of a gap as possible between them, called a margin. A SVM constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. With our training set as an input, the SVM finds the hyperplane such that each point is correctly classified and the hyperplane is maximally far from the closest points. The name "support vector" comes from points on the margin between the hyerplane and the nearest data points, which are called support vectors. The SVM looks for a parameter vector a that, again, maximizes the distance between the hyperplane and every training point. In essence, it is an optimization problem:

$$\text{Minimize } \frac{1}{2}a * a$$

$$y(a * x_i + b) >= 1 \ ,$$

where y is the class label $(-1, 1)$ for negative and positive.

Once the SVM is built, classification of new tweets simply involves determining which side of the hyperplane that they fall on. In our case, there are only two classes, so there is no need to go to a non-linear classifier.

Figure 3 displays the sentiment classification of all gathered tweets, concerning seven of the most popular tech companies, from 14/6 until 22/6 using the following formula to normalize their positive sentiment score over the total number of tweets per day:

$$NormalizedPositiveScore_i = \frac{PositiveTweets_i}{TotalTweets_i}$$

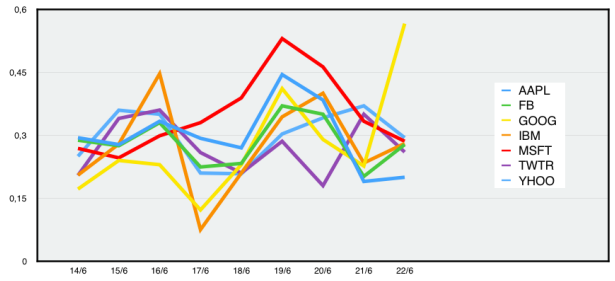

Figure 3: Normalized Positive Score from 14/6 to 22/6

Classifier Evaluation

In order to accomplish our ultimate goal, which was to utilize Twitter sentiment scores for predicting stock market's movements, we trained and tested each of our classifiers on a particular subset of our tweet corpus. So as to calculate accuracy, we used the k-fold cross validation method. In this method, we train on all days upto a specific day and test for the next k days. For the purpose of our analysis, we used 7-fold cross validation and found the average accuracy. k-fold cross validation splits the training dataset into k smaller datasets. One of these subsets is left out and used as a test set to measure accuracy and the other $k-1$ subsets (in our case 6), are used to train the classifier. This procedure is repeated k times, every time for a different data partition.

Firstly, we measure accuracy and secondly, precision and recall values for each of the labels, "positive", "negative" and "neutral". Precision is measured as the number of true positive tweets over all positive tweets:

$$Precision = \frac{tp}{tp+fp} \ ,$$

where "tp" stands for true positive tweets and "fp" for false positive tweets. On the other hand, Recall is the true positive rate of the classifier and it's calculated as follows:

$$Recall = \frac{tp}{tp+fn} \ ,$$

where "fn" stands for false negative tweets.

Machine Learning Evaluation metrics		
Algorithm	Naive Bayes	SVM
Accuracy	0.80609	0.79308
Precision	pos: 0.8	pos: 0.7727
	neg: 0.62	neg: 0.4545
	neut: 0.8421	neut: 0.8125
Recall	pos: 0.6	pos: 0.5666
	neg: 0.9473	neg: 0.8333
	neut: 0.7441	neut: 0.8666
F-measure	pos: 0.6857	pos: 0.6538
	neg: 0.75	neg: 0.5882
	neut: 0.79	neut: 0.8387

Table 1: Average effectiveness of machine learning techniques for sentiment classification.

7. STOCK MARKET PREDICTION

After training our classifier, we decided that an interesting application would be to look at the correlation between tweet sentiment and stock market prices. We focused on the top-16 technology stocks according to Yahoo! Finance. After we completed the Data Preprocessing, both in Twitter data and in stock data, as described in Section 4, we created a feature matrix which contains the following features:

- percentage positive sentiment score
- percentage negative sentiment score
- percentage neutral sentiment score
- close price
- HLPCT
- PCTchange
- volume

Then we applied SVM in order to achieve a prediction about future stock movements. Afterwards, we compared our results with the upcoming stock changes. Conclusively, we achieved an average accuracy of 87% concerning correct stock's movement prediction.

In terms of the predicted closing price we accomplish prediction errors under 10%. The figure below shows our prediction results on 23/6 for the 16 most popular technology stocks according to Yahoo! Finance.

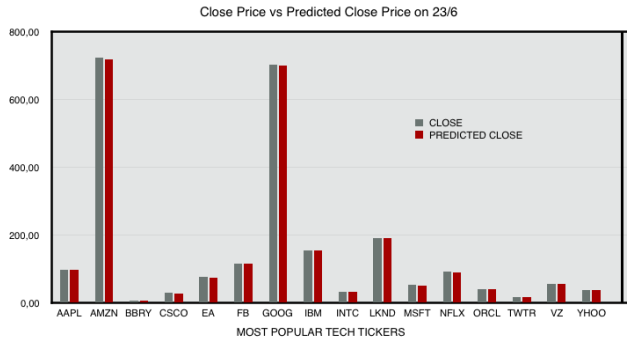

Figure 4: Close Price vs Predicted Close Price on 23/6

In the extra figure below it is clarified that the largest error is 6.29% in the case of "Blackberry" (BBRY) and that nine out sixteen errors are below 1%. The average prediction error is 1.668%.

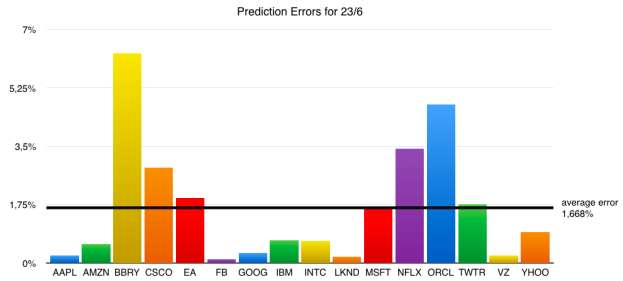

Figure 5: Prediction Errors for 23/6

8. CONCLUSIONS AND FUTURE WORK

In this paper, we investigated whether public sentiment, as measured from tweets, is correlated or even predictive of stock values and specifically for 16 of the most popular tech companies according to Yahoo! Finance. Our results show that changes in the public sentiment can affect the stock market. That means that we can indeed predict the stock market with high chances.

Furthermore, it is worth mentioning that our analysis does not take into consideration many factors. First of all, our dataset does not really extract the real public sentiment, it only considers the twitter using, english speaking people. Secondly, the bigger the dataset is, the better the prediction but at the same time the problem gets more complicated.

There are many areas in which this work could be expanded in the future. With a longer period of time and more resources, there is much potential in the field. If it is possible, we would want to collect data over the course of a few years, both from Twitter and the stock market. In addition we could investigate intraday stock changes in order to make our prediction more accurate. Finally, in the future we could create a stock lexicon based on the most common words used.

9. REFERENCES

[1] W. Antweiler and M. Frank. Do US stock markets typically overreact to corporate news stories? *Working Paper*, (1998):1–22, 2006.

[2] J. Bollen and H. Mao. Twitter mood as a stock market predictor. *Computer*, 44(10):91–94, 2011.

[3] a. Mittal and a. Goel. Stock Prediction Using Twitter Sentiment Analysis. *Tomx.Inf.Elte.Hu*, (June), 2012.

[4] F. Å. Nielsen. A new ANEW: evaluation of a word list for sentiment analysis in microblogs. In M. Rowe, M. Stankovic, A.-S. Dadzie, and M. Hardey, editors, *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, volume 718 of *CEUR Workshop Proceedings*, pages 93–98, May 2011.

[5] A. Pak and P. Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. *Lrec*, pages 1320–1326, 2010.

[6] T. O. Sprenger, A. Tumasjan, P. G. Sandner, and I. M. Welpe. Tweets and trades: The information content of stock microblogs. *European Financial Management*, 20(5):926–957, 2014.

[7] P. D. Wysocki. Cheap talk on the web The determinants of postings on stock message boards. 1998.