Twitter Stock Bot

John Matthew Fong
The University of Texas at Austin
jmfong@cs.utexas.edu

Hassaan Markhiani
The University of Texas at Austin
hassaan@cs.utexas.edu

Abstract

The stock market is influenced by public information and opinion, and Twitter is an accessible communication tool we can use to gauge public opinion. In this paper, we utilize Twitter to construct a tool that provides information about the stock market. We incorporate many natural language processing techniques to provide feedback about a given company. More specifically, we use sentiment analysis to determine a company's public opinion and build a classifier to predict the returns of a stock based on article snippets from recent New York Times articles. Our results show that our sentiment analyzer achieves an accuracy of 49.43% and 58.80% and our classifier achieves an accuracy of 45.66%. This indicates that there is a correlation with news article snippets and stock returns, and that it is possible to predict expected returns based recent articles.

1 Introduction

The stock market allows for the trading of company stocks at agreed prices. By buying and selling shares at different prices, it is possible to earn a profit, so there is an entire industry around helping investors optimize and maximize their profits. An important aspect of this industry is being able to predict the movements in the stock market, and any bit of information that offers a competitive edge to an investor is relevant.

Since investors determine the price of a stock based on publicly available information about the company, public information has a significant impact on the stock market. Thus, it is reasonable to conclude that financial news articles and twitter's opinion may have an influence on the movement of certain stocks. We wanted to build a tool for investors that could provide this information, so we decided to implement a bot that provides relevant information about a company. Given a tweet, the bot provides information for the company name or stock exchange symbol mentioned in the tweet. The bot provides the last price of the company's stock, an estimate of the outlook of the stock price, public opinion of the company on Twitter, and a link to the Yahoo! Finance page for the company.

The stock information is obtained through Yahoo!'s Yahoo Query Language API, and the link in the bot's tweet is shortened using Bit.ly's API. The stock outlook is determined based on snippets from articles related to the company retrieved from the New York Times using their API. The articles are processed by our classifier to determine the stock's outlook. The sentiment analysis portion of the bot is based on a few of Liu's ideas and approaches (Liu, 2010). The goal of the project was to build a bot that can provide helpful information to an investor, as a tool, so our bot also provides helpful responses in ambiguous situations.

Since the sentiment analysis and stock outlook prediction are major components of our project, we evaluated their performance by testing on untrained datasets. We tested our sentiment analysis method on a set of tweets from the 2008 presidential debate (Shamma et al., 2009; Diakopoulos and Shamma, 2010) and the Stanford Twitter Sentiment dataset (Go et al., 2009). Our results show that our sentiment analysis method is able to achieve an accuracy of 49.43% and 58.80% respectively. For our stock outlook prediction, we constructed 4 sets of training and testing corpora. The set of news articles is the same across the different corpora and contain the 10 most recent articles about the NASDAQ, NSYE, and AMEX companies from the New York Times. The only difference is that the labels are based on returns over different time periods (1, 3, 5, and 7 days). Our

classifier is able to achieve an accuracy of 45.66% on the testset for the 7 day returns, which was the highest.

2 Parsing Tweets

When our bot receives a status (tweet), it first tokenizes the text of the status. In the last iteration of our bot, we used a simple tokenizer that left in characters like punctuation. However, we switched to using a custom tokenizer that removes and splits on all characters that aren't letters or numbers because we found this lead to better performance. Because we are only dealing with disambiguating the names of companies, it makes sense to remove characters that don't commonly appear in the names of companies. Question marks, exclamation points, apostrophes and various other characters are very rare in company names, but our methods are robust enough to correctly handle disambiguation even when these characters do appear. We also remove all stopwords from the tweet text because these words are not important when determining which company a tweet refers to.

After tokenizing, we search an inverted index for companies that may have been mentioned in the tweet. For each word mentioned in the tweet, every company with the same word in its name gets a point. If one company gets the most points, that company is determined to be the company that was asked about in the tweet. Alternatively, there are several states that will cause our bot to be "confused" and ask for more information from the user. If multiple companies are tied for the top score, then we select the words in the tweet that are contained in these companies' names, tell the user that these words led to confusion and ask for clarification. However, if no companies are found we consider all words in the tweet to be confusing and once again ask the user for clarification. A special case arises when the tweet only contains stopwords (all of which are removed), which leads us to select one of several random replies to the user.

2.1 Inverted Index

We use an inverted index as our main method of disambiguating company names. The inverted index is a map of words to a set of stock symbols whose corresponding company names contain that word, illustrated in Figure 1. We build the index by reading two files with data about companies

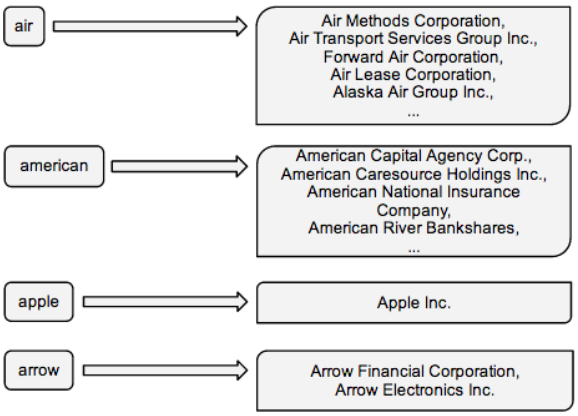

Figure 1: Illustration of our inverted index. Note that company names are shown for clarity, but the index actually stores symbols such as AAPL (Apple, Inc.).

from the NYSE and NASDAQ stock exchanges. Similar to parsing tweets, the company names are tokenized using our alphanumeric tokenizer and the symbol is added to the set that corresponds to each word. This is convenient, efficient and effective because words like "Apple," "Microsoft," and "Google" have only one entry, as do many other company names. When a word has many associated companies, performance is not hurt because mentioning these words alone would not typically be enough information to disambiguate the company name in any circumstance. Our counting method yields a similar result to taking the union of the sets of companies related to words in the tweet, and usually leads to a unique result when a full company name is mentioned.

2.2 Special Syntax

In addition to our typical behavior that involves parsing and responding to tweets, there is a special syntax that our bot responds to. When a user tweets with the format "UPDATE (SYMBOL): Company name", the bot takes the stock symbol and company name that were mentioned and updates its inverted index. Using this ability we can improve the bot's behavior by adding more companies, or different names for companies. For example, without an update the bot cannot find the company "American Airlines." This is because the company is owned by the AMR Corporation, though many users may not know this. Using the special syntax we can update the bot with the more colloquial and commonly used name.

3 Predicting Stock Return

In order to predict stock return, we analyzed articles from the New York Times regarding the company. We began by building a corpus of article snippets from the New York Times. We used the New York Times API to query for article snippets for each company in the NYSE and NASDAQ. The API returned the 10 most recent articles, but since it is possible for some articles to be unrelated to stock price, we filtered the set to only contain articles from the "Business Day," "Technology," and "Your Money" sections. Finally, we obtained the average returns achieved over the varying time periods following the article's release from the companies related to the article. In total, we were able to collect 5,196 articles.

The corpus was then split into 90% (4,676 articles) and 10% (520 articles) for our training and testing dataset respectively. Using the dataset, a classifier was produced to predict a stock's expected returns, similar to the methods performed previously (O'Flaherty and Folsom, 2012). Since it is difficult to directly predict the returns, the classifier predicts one of three labels (Good, OK, and Bad). The thresholds of the labels are described in Table 1.

The classifier uses L2-regularized logistic regression with the following features for each document: tokenized and stemmed word counts, polarity of the article, extracted company names, and a dollar sign counter. We used the Twokenizer from Nak and the Porter Stemmer from Chalk for tokenization and stemming respectively. The polarity of the article is determined using the same technique described in Section 4. The inverted index method used to extract company names from tweets (Section 2.1) is also used to extract possible company mentions. However, rather than force the result of company name disambiguation to be unique, we use the entire list of company names occurring in the article as a feature. This is necessary because we cannot ask the articles for clarification, like we do with users on Twitter, and we found that the use of a list of possible company names improved our results more than only using the feature when a unique name occurs.

To predict the stock outlook, the New York Times API was used to obtain the most recent articles snippets about the company. In our case, any article published within a week is considered recent. The classifier then identifies the most probable label for each of the recent and relevant articles. Finally, the most occurring label across the articles is used as the stock outlook. If the API could not find any recent and relevant articles, then we use the simple metric described in Equation 1, where p_c is the current price of the stock, p_l is the year's lowest price and p_h is the year's highest price. When calculating stock outlook in this way, it is calculated to be Good, OK, or Bad based on where the current price lies in the range of the stock's 52 week high and low. If the current price is above 70% of the range, then the outlook is Good. If the current price is between 30% - 70% of the range, then the outlook is OK. If the current price is below 30% of the range, then the outlook is Bad.

Label	Expected Return
Good	> 0.005
OK	$\leq 0.005, > -0.005$
Bad	≤ -0.005

Table 1: Stock Outlook Classifier Labels.

$$outlook = \frac{p_c - p_l}{p_h - p_l} \quad (1)$$

4 Sentiment Analysis

Determining the sentiment of tweets and business articles is a crucial portion of our bot's behavior. Analyzing the polarity of tweets on Twitter is the sole way we determine if public opinion of a company is good or bad. We also use sentiment analysis as a key feature in our classifier that decides whether a business article indicates whether a stock will go up or down in the near future.

Rather than perform a simple sentiment analysis by counting the number of positive and negative words, we use a slightly more complicated method that utilizes bigrams in the text and negates polarity where it is appropriate. We first generated a list of "negation" words that consists of the words "no" and "not" as well as all contractions that end in "n't". Because it is common practice on Twitter to leave out apostrophes, we have also included duplicates of these negative contractions with apostrophes removed. For example, "dont" and "cant" are considered negations words. When analyzing bigrams of the text, if the first word is one of our negation words, we reverse the polarity of the second word. Words that are neither

positive nor negative are not affected. Our sentiment metric is defined in Equation 2, where "Pos" is the number of words judged to be positive and "Neg" is the number of words judged to be negative (Flinchbaugh and Latimer, 2012).

$$sentiment = \frac{Pos - Neg}{Pos + Neg} \quad (2)$$

5 Bot Behavior

Conversations we have had with the bot can be seen in Figures 2 - 6. When asked a question about a single company, the bot can successfully respond with information about that company. The format of tweets containing company information is shown below.

If multiple companies or no companies are mentioned, the bot responds asking for clarification with the stock symbol. An example of the inverted index being updated is also shown in Figure 6.

$$@Twitter\_Handle\ Company\_Name$$
$$(Company\_Symbol), Price : Stock\_Price,$$
$$Outlook : (Good|OK|Bad),$$
$$Opinion : (Good|OK|Bad),$$
$$Info : Yahoo\_Finance\_Link$$

6 Evaluation Methodology

Since the sentiment analysis and stock outlook prediction are major components of our project, we evaluated their performance by testing on untrained datasets. We tested our sentiment analysis method on a set of tweets from the 2008 debate (Shamma et al., 2009; Diakopoulos and Shamma, 2010) and the Stanford Twitter Sentiment dataset (Go et al., 2009). For our stock outlook prediction, we constructed 4 sets of training and testing corpora. The set of news articles is the same across the different corpora and contain the 10 most recent articles about the NASDAQ, NSYE, and AMEX companies from the New York Times. The original set of articles obtained from the New York Times contained 5,196 articles, which was split into a 90% training set (4,676 articles) and a 10% testing set (520 articles). The only difference between the 4 datasets is that the labels are based on returns over different time periods (1, 3, 5, and 7 days). We then tested our trained classifier on the test sets. Our goal was to get a measure for the accuracy of our two systems.

Figure 2: Normal Behavior

Figure 3: Response when multiple companies may have been mentioned

Figure 4: Response when no companies can be found

Figure 5: Response when the tweet contains only stop words

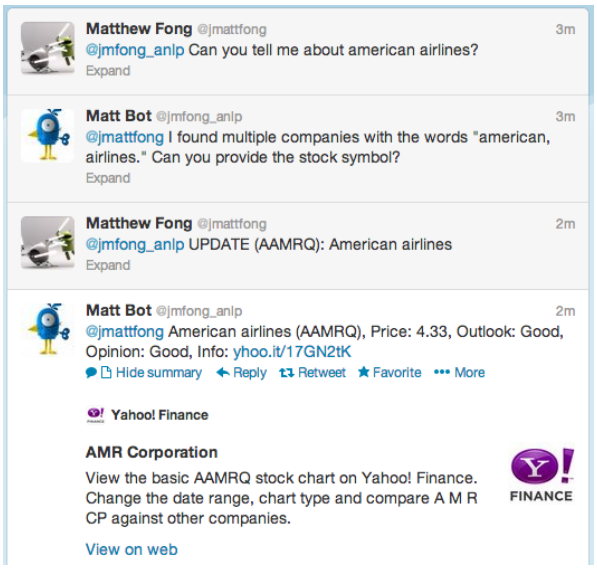

Figure 6: A conversation illustrating the UPDATE special syntax

7 Results and Discussion

Our results turned out well; both of our systems significantly outperformed the random baseline. Our sentiment analysis system was able to achieve an accuracy of 49.43% and 58.80%. Our system achieved decent f-scores across all categories as well (Table 6). Although our lexicon based approach worked fairly well, machine learning algorithms could have achieved better results (Go et al., 2009). We attempted to train a classifier for our sentiment analysis portion, but we were not able to construct an accurate classifier based on the data available. Our lexicon based approach achieved higher accuracies than classifiers trained one set and tested against the other, i.e. trained on the debate dataset and tested on the Stanford Twitter Sentiment dataset and vice-versa.

Our results for the stock outlook classifier were interesting. We achieved the highest accuracy predicting the 7 day returns, which was a bit surprising. We were expecting the 1 day returns to achieve the highest accuracy because we believed that the effect of an article might be short term. Our 7 day return stock outlook classifier was able to achieve an accuracy of 45.66%, with an average error of 0.8959. An error below one means that on average, well performing stocks will not be classified as bad and poor performing stocks will not be classified as good. According to Table 3, our classifier does a good job of predicting Good returns, a decent job of predicting Bad returns, and a terrible job of predicting OK returns. This works out well for us because our users would rely mostly on Good and Bad predictions, while ignoring OK predictions. Unfortunately, these results could have been better (O'Flaherty and Folsom, 2012). Currently, during training of our classifier, our system builds the list of an article's mentioned companies based on the New York Times search results. A better metric might be to only consider companies that were explicitly mentioned in the article rather than trust the results of the API call that might have returned only articles indirectly related to the company. The returns being predicted are based on the mentioned companies, so our current system might incorporate returns of stocks that we not talked about in the article. Using the stricter approach might produce better results.

N (days)	Accuracy
1	41.62
3	41.62
5	43.35
7	45.66

Table 2: Overall accuracy for predicting the N day stock return based on analysis of business articles.

54	124	5	183	Bad
59	180	4	243	Good
18	72	3	93	OK
131	376	12		
Bad	Good	OK		

Table 3: Confusion matrix for predicting the 7 day stock return based on analysis of business articles. Columns give predicted counts. Rows give gold counts.

8 Conclusion

We set out to build a useful tool for investors that would provide real time information about both the future performance of stocks and the public opinion of stocks with natural language processing techniques. Our bot has the ability to disambiguate company names when specific questions are asked and can be updated when its knowledge of company names is incomplete. The bot can therefore provide information about a company when asked questions about it in plain English. By using a classifier trained on a corpus of past news articles, we were able to predict 7 day returns for stocks 45.66% accuracy, with f-scores for predicting good and bad performance much higher than a random selector could perform. We also used Twitter to gauge the real time opinion of users about a company to provide valuable information about public opinion. In tests of our accuracy predicting sentiment, we achieved a respectable 49.43% and 58.80% accuracy when testing on two different data sets, labelled tweets from the presidential debate and the Stanford Twitter sentiment data set. While the bot is likely not sophisticated enough to be the sole tool used by investors, we believe that our results indicate it could be useful as a filter when deciding which stocks to invest in by providing a quick analysis that might otherwise take users a large amount of work to duplicate.

Precision	Recall	F-score	
41.22	29.51	34.39	Bad
47.87	74.07	58.16	Good
25.00	3.23	5.71	OK
38.03	35.60	32.76	Average

Table 4: Precision, recall, and f-score for predicting the 7 day stock return based on analysis of business articles.

136	173	63	372	negative
21	135	41	197	neutral
9	95	122	226	positive
166 negative	403 neutral	226 positive		

Table 5: Confusion matrix for estimating sentiment of tweets from the 2008 presidential debate. Columns give predicted counts. Rows give gold counts.

9 Code

All of our code is accessible from github.com. Please keep in mind that API keys are required to run the project but are not provided. The format of the ".properties" files is provided on the project's wiki page. Link:
$https://github.com/hassaanm/tshrdlu$

References

Rowland O'Flaherty and Woody Folsom. 2012. *Predicting Stock Trends Using Natural Language Processing of Headlines*.

Bo Pang and Lillian Lee. 2008. *Opinion Mining and Sentiment Analysis*. Foundations and Trends in Information Retrieval Vol. 2, Nos. 12.

Bing Liu. 2012. *Sentiment Analysis and Subjectivity*. Handbook of Natural Language Processing, Second Edition.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. *Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis*. IJCAI-07: 1606-1611.

Silviu Cucerzan. 2007. *Large-Scale Named Entity Disambiguation Based on Wikipedia Data*. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning: 708-716.

Gyozo Gidofalvi. 2001. *Using News Articles to Predict Stock Price Movements*. Department of Computer Science and Engineering, University of California.

Precision	Recall	F-score	
81.93	36.56	50.56	negative
33.50	68.53	45.00	neutral
53.98	53.98	53.98	positive
56.47	53.02	49.85	Average

Table 6: Precision, recall, and f-score for estimating sentiment of tweets from the 2008 presidential debate.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Hongjun Lu. 2005. *The Predicting Power of Textual Information on Financial Markets*. IEEE Intelligent Informatics Bulletin Vol 5 No 1.

Wenbin Zhang and Steven Skiena. 2010. *Trading Strategies to Exploit Blog and News Sentiment*. In Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media.

Ben Frisbee. 2010. *The Predictive Power of Financial Blogs*. Department of Economics, Haverford College.

Robert Schumaker. 2010. *An Analysis of Verbs in Financial News Articles and their Impact on Stock Price*. In Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media: 34.

David A. Shamma, Lyndon Kennedy, and Elizabeth F. Churchill. 2009. *Tweet the Debates: Understanding Community Annotation of Uncollected Sources*. In Proceedings of the First SIGMM Workshop on Social Media: 3-10.

Nicholas A. Diakopoulos and David A. Shamma 2010. *Characterizing Debate Performance via Aggregated Twitter Sentiment*. In Proceedings of the 28th International Conference on Human Factors in Computing Systems: 1195-1198.

Alec Go, Richa Bhayani, and Lei Huang. 2009. *Twitter Sentiment Classication using Distant Supervision*. Stanford University.

Anne Flinchbaugh and Eric Latimer 2012. *ReelTalk: Project Phase 5 - An interactive sentiment analysis application*. The University of Texas at Austin.