FINAL REPORT – STOCK MONEY MINERS (GROUP #8)

Austin Vaday
UCLA Student
(104566193)

Ilya Tabriz
UCLA Student
(604675793)

Nilesh Gupta
UCLA Student
(604489201)

ABSTRACT

Stock market prediction is a commonly cited problem of interest. While quantitative analysis can provide useful information about a stock, the movements of a stock are rarely ever based solely on the numbers; reputation and public perception play a great role in the value of a company's stock. In theory, by modeling a popular public forum (Twitter) as a real time reflection of the public and mining for tweets relevant to a given company, one should be able to gauge that company's standing in the public's eyes and use this info to predict its stock. We do so here by extracting all tweets containing a search query such as a company's index and performing sentiment analysis on the aggregate body of tweets we receive. Using Word2vec to represent relationships between words and their sentiments in these tweets, we can accurately establish the public image of company at a given time. We then compare this image to the company's near-term stock prices to see if there is a relation. Based on preliminary results and some related works, we should find that a relatively strong correlation exists between public sentiment and stock prediction.

1. INTRODUCTION

In today's day and age, stock prices fluctuate every second, causing flurry and cheers all around the world. In the United States in particular, Individuals on Wall Street flock to purchase and sell stocks in an extremely demanding and high-paced environment. Given that stocks rise one day, and tank on other days, how do we accurately predict the value of a particular stock? Is there a sense of randomization, or is there a way to properly formulate and accurately predict such stock fluctuations? Traditional approaches for stock prediction involve analyzing the past stock prices, but our goal and focus here is to utilize current and present reactions to stock prices on Twitter. With this live data on social media, we seek to not only predict, but formulate a model through the data we gather.

2. PROBLEM

Data mining, the practice of examining large data sets and attempting to formulate new information and trends, can be used as a tool to help us formalize our problem. Some functionalities of data mining include "discovery of concept or class descriptions, associations and correlations, classification, prediction, clustering, trend analysis, outlier and deviation analysis, and similarity analysis" (reference 1). Our problem formalization in our quest to predict and analyze stock data for the future is as follows: How can we use data from social media platforms such as twitter to predict which way a stock will most likely move in the short term?

3. DATA PREPARATION AND PREPROCESSING

In order to crawl twitter data, we need to find a mechanism or server that hosts our crawl code. One option would be to create an AWS (Amazon Web Services) EC2 instance that hosts a python script. This python script would be invoked at a custom frequency set up through a Cron Job. The python script, whenever invoked, would "crawl" the necessary twitter data and store the results either in a local or remote datastore. Another option would be to create an AWS lambda function (written in python) that can be invoked manually, utilizing the same code on our EC2 instance but instead being invoked on-demand rather than periodically on a server. The benefit of using a lambda function is that one may pay per script invocation rather than for an entire server that is not being utilized very often. And the final option would be to simply have a python script that would be executed on-demand from a local environment. This is the approach we utilized, as it is simple and allows us to focus on our data mining approach rather than dwelling on the technological infrastructure.

Once we have our infrastructure set, our next focus is on the data mining approach. We currently have a script that will use the tweepy library from Python to query the twitter database for all tweets containing a specified string such as "AAPL", "iPhone X", and so forth. We can add various filters to the query to narrow down our search results, such as filtering for tweets written in English for example. We chose to create a script that structures various queries and stores all unique tweets into a local CSV file, namely tweet_training_tesla.csv. The queries we ran are shown as follows:

```
query_list = [("Tesla", "stock", "dow", "win"),
        ("Tesla", "stock", "lose"),
        ("Tesla", "stock", "win"),
        ("Tesla", "stock", "up"),
        ("Tesla", "stock", "down")]
```

Where the each query is a tuple object that searches for tweets that best fit the corresponding request. I.e. ("Tesla", "stock", "lose"), will return the tweets that are most likely associated with negative tweets that indicate Tesla stock being a loss.

We also needed to generate additional tweets to run and test our algorithms on. For this, we queried for nearly 700 tweets that are related to either "Tesla" or "stock": ("Tesla", "stock"). These tweets are stored in twets_generated_tesla.py and its usage will be discussed later.

Once we have these tweets, the next step is to perform a text preprocessing on our tweets as seen in stop_word.py. For our data preprocessing, we elaborate on three stages. First, we perform a Special Character Removal, which involves regular expression pattern matching to remove hashtags and other irrelevant parts of the tweet using the tweetPreprocessor python package. The second, Stopword Removal, involves removing various words that do not contribute to our analysis of the tweet user's emotion. And finally, we perform Tokenization, which involves creating a list of individual words used in each tweet.

To get the stock data, we tried a few different approaches. The first was using the Google and Yahoo Finance APIs in Python. When we found out these were deprecated, we looked into other APIs but found that none offered historical stock data (without charge). At this point, we almost settled on hand copying 200 cells of stock data into a csv. Fortunately, we stumbled upon http://finance.google.com/finance/historical?q=, a stock price database offering free web request queries. By feeding the query shown below into a requests.get call,

stock_query="http://finance.google.com/finance/historical?q=TSLA&startdate=11+28+2017&enddate=12+11+2017&output=csv"

we were able to obtain daily stock close, open, high, low and volume quantities for $TSLA over our target time period. Since the target was to analyze how tweet sentiments may affect stock prices, we added a 'Change' column representing the change in the stock's price some number of days down the line.

One thing to note is while this method automated the process of collecting stock data, it did not carry hourly stock prices. Because the tweepy API only allowed us twitter data from the past week, we initially thought we come compensate for this shortstack by getting more precise stock data and comparing it hour by hour. This approach could not be sustained since (a) Python has no free API with hourly stock data (b) the unpredictability of our data would have increased tremendously with an hour by hour price change comparison. However, in the future this may be a methodology worth further investigation.

Once we collected both the stock data and the twitter data with classifications, the only thing left to do was to merge the two datasets. First, each csv was fed into a dataframe from the panda library in python. Next, the date strings for both the tweets and stock prices were converted to datetime objects to make sure each set had the same formatting. Then grouping the sentiment data by date and classification, we were able to get a count of the positive, negative, and neutral tweets per day for the weeks in the tweet collection period. Note that we drop the tweets at this point since we have no need for them, only their classifications. Finally, we merged the two datasets on their 'Date's to obtain a final Dataframe containing the stock close, change in stock price, and the aggregate tweet classifications for each date in the time span for which we had stock data (no weekends).

4. METHODS DESCRIPTION (DETAILED STEPS)

Once we have completed the data preprocessing stage, we proceed to perform sentiment analysis and attempt to classify tweets as either Positive, Negative, or Neutral using the classification techniques we have learned in class.

We initially thought about utilizing the scikit-learn python library to aide in our classification and sentiment analysis. This involves performing a feature extraction, and can be accomplished by building a Word2Vec model with Twitter data, which simply transforms words into vectors and allows us to find the top N closest words given a particular word of interest. This is one mechanism for analyzing how positive, negative, or neutral a particular tweet is. However, we found this too complex for the sake of our project, and we were advised to construct the training set manually.

We first created a training set, labeling over 200 tweets as Positive, Negative, or Neutral from our own observations (see tweet_training_tesla.csv). Then, we used Python library called Tweepy to connect to Twitter to download tweets related to 2 keywords: "Tesla","stock". we wrote a script called stock_crawler.py that fetched the data from twitter, Initially data was formatted in json, we convert data into to a CSV format that includes sentiment,Timestamp and tweet. training tweets were trained manually and classified as positive or negative and neutral. In tweet_class.py we preprocess the data, preprocess functions were defined at stop_word.py we removed stop words using list of words in sklearn.Feature_extraction.stop_words, we remove multiple characters inside one word (ex: appple would become apple) and we removed symbols and any character that is not in english alphabet or a number. Then using nltk algorithm we evaluated feature vector and using feature vector we created set of features that contain a large number of specific words. our Feature extraction functions behaved like tinted glasses, highlighting the emotions (pos,neg and neutral) in our data.

Our classifier mostly relies on highlighted words when determining how to classify the inputs. our classifier made its decisions based on information about which of the common words (if any) a given tweet contains.we evaluated our features set to have not too many featured word, because then algorithm would have a higher chance of relying on idiosyncrasies of our training data that don't generalize well to new examples and avoid overfitting.

Using error analysis we optimized our training data. then we used nlk module and our features set to Classify new tweets using NaiveBayesClassifier. Using nlk we found most informative features from our manually trained data. it is easy to see that "downgraded" was more seen in tweets that were classified as -1 and "rising" was more seen in tweets that were classified as 1 and "think" was more seen in tweets that were classified as 0. the accuracy of algorithm over our manually trained data set was 0.75 which is higher than we expected to be. We used our algorithm to classify set of tweets from one week to use one week of tweets and previous stock price for price prediction . some of the words form Most Informative Features from test data was "falls" which was was more seen in tweets that were classified as -1 and "electric" which was was more seen in tweets that were classified as 1 and accuracy was 0.85 which was high. using more trained tweets which includes more general words could have helped up to have more accurate algorithm. List of Informative Features from training data and test data is provided below.

4.1 Manually Trained Data

Most Informative Features

giants = True	0 : -1	= 39.2 : 1.0
horse = True	0 : 1	= 31.5 : 1.0
tech = True	0 : 1	= 31.5 : 1.0
demand = True	-1 : 1	= 19.0 : 1.0
know = True	0 : 1	= 14.6 : 1.0
cars = True	0 : -1	= 14.3 : 1.0
like = True	0 : -1	= 13.5 : 1.0
investors = True	0 : -1	= 12.4 : 1.0
buying = True	0 : 1	= 10.0 : 1.0
make = True	0 : -1	= 8.6 : 1.0
crazy = True	0 : -1	= 8.6 : 1.0
falls = True	0 : -1	= 8.6 : 1.0

```
fine = True   0 : -1  =  8.6 : 1.0
investments = True   0 : -1  =  8.2 : 1.0
going = True   0 : 1   =  6.9 : 1.0
said = True   0 : 1   =  6.9 : 1.0
model = True   0 : -1  =  6.7 : 1.0
world = True   0 : -1  =  6.7 : 1.0
money = True   0 : -1  =  6.3 : 1.0
electric = True   1 : 0   =  5.6 : 1.0
```

Accuracy: 0.8587731811697575

4.2 Data classified using classifier

Most Informative Features

```
downgraded = True  -1 : 0  =  13.5 : 1.0
demand = True  -1 : 1  =  9.4 : 1.0
starting = True  -1 : 0  =  7.7 : 1.0
signs = True  -1 : 0  =  7.7 : 1.0
sell = True  -1 : 0  =  7.2 : 1.0
flag = True  -1 : 0  =  6.9 : 1.0
order = True   1 : 0  =  5.9 : 1.0
semis = True   1 : 0  =  5.9 : 1.0
places = True   1 : 0  =  5.9 : 1.0
rising = True   1 : 0  =  5.9 : 1.0
hold = True  -1 : 0  =  4.8 : 1.0
electric = True   1 : 0  =  4.1 : 1.0
tesla = True   0 : -1  =  3.3 : 1.0
shares = True  -1 : 0  =  3.3 : 1.0
customer = True  -1 : 0  =  3.3 : 1.0
lagging = True  -1 : 0  =  3.3 : 1.0
going = True  -1 : 1  =  3.0 : 1.0
think = True   0 : -1  =  2.7 : 1.0
analyst = True  -1 : 0  =  2.6 : 1.0
```

Accuracy: 0.7570621468926554

5. EXPERIMENTS DESIGN AND EVALUATION

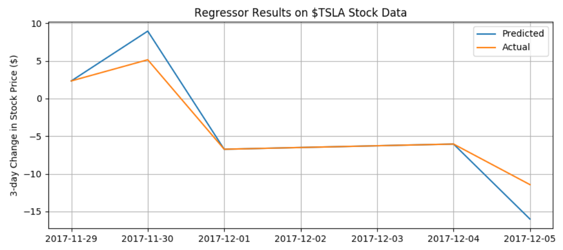

Figure 1: Scatterplot of Regression results using LinearRegressor from sklearn and 3 days as stock change interval. `Correlation (r2 score): 0.742739`

The above graph shows the results of the Linear Regressor over the time period of our tweet collection. It uses the number of positive, negative, and neutral tweets obtained from our sentiment classifier to make predictions on stock movement in 3 days. We experimented with different values for days and found that twitter sentiment about a company seems to be most reflected in that company's stock 3 days from when the sentiment was recorded. This is most likely due to the inherent inertia in the idea of observing slow and relatively constant patterns in the stock market by realtime, often reactive trends on twitter.

The final score of the regressor was about 74% with these parameters. Due to the scarcity of data afforded by twitter, we were not able to extensively test this on as long of a period as we would have liked. However, this result does not disqualify the idea that twitter sentiment may serve as a promising indicator of a company's public standing and as a consequence, its stock price.

6. RELATED WORK

TASK	PEOPLE
1. Collecting, Preprocessing, and Storing Tweet	Austin Vaday, Ilya Tabriz
2. Manual Classification of Data	Austin Vaday, Nilesh Gupta
3. Implementing text Classification Algorithm	Ilya Tabriz
4. Implementing Stock Price API Fetch	Nilesh Gupta
5. Implementing Regression Algorithm	Nilesh Gupta
6. Evaluating and comparing algorithms	Ilya Tabriz, Nilesh Gupta
7. Writing report	Austin Vaday, Ilya Tabriz, Nilesh Gupta

7. CONCLUSION

Through our experiment, we have analyzed how accurately we can predict the value of a particular stock. We discovered a way to properly formulate and predict stock fluctuations, even though we had a small data set limitation to begin with. Our classifiers allowed us to classify new tweets as positive, neutral, or negative given a training size of a few hundred tweets. And as a result - our linear regressor achieved 74% accuracy, allowing for us to predict such fluctuations in stock prices in a reasonable way. Overall, this project allowed for us to not only demonstrate, but also strengthen our data mining and implementation skills as future engineers.

8. REFERENCES

[1] http://ieeexplore.ieee.org/document/4476641/?part=1

[2] http://www.acit2k.org/ACIT/2013Proceedings/163.pdf

[3] http://scikit-learn.org/stable/

[4] https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/

[5] https://www.tutorialspoint.com/data_mining/dm_classification_prediction.htm

[6] https://arxiv.org/pdf/1610.09225.pdf