Stock Market Manipulation through Influential Tweets

April 16, 2021

Authors:
Bruce Chidley
Alice Petrov
Kyle Maede
Matthew Rezkalla

Abstract

Historically, SEC charges have demonstrated that CEOs have the potential to influence their company stock through social media posts, creating a controversial and inadequate regulatory landscape. In this paper, we evaluate the potential relationship between the sentiments of social media posts made by CEOs and their respective stock movements. In particular, we collect a set of 13,006 tweets spanning a timeframe of two years in order to investigate the impact these posts may have. A key challenge in this domain is producing consistent results and insights with inconsistent data. We propose multiple machine learning models in an attempt to predict stock movement based on the sentiment of tweets made by the stock's respective CEO, parallel to a time-series. In particular, we analyze and evaluate the performance of predictive models in order to investigate the potential correlation between CEO tweets and their corresponding stock. Positive, but unreliable accuracies, were achieved through both a Bi-directional Recurrent Neural Network and a Naive Bayes model. We then analyze a Multilayer Perceptron Model on specific companies, assessing the possibility of better performance on a more specific prediction task. This approach shows promising results, and further investigation on a larger dataset is required in order to confirm this model's performance.

Introduction

The 2008 financial crisis set off a domino effect that laid substantial socio-economic damage across the entire world, which took many countries around half a decade to fully recover from. It began in the United States, resulting in the loss of approximately 5.5 million jobs [8]. This was a direct result of market manipulation and general recklessness of the poorly regulated financial industry [9]. Nonetheless, it was a historical event that demonstrated the significant weighting financial securities have on our livelihood, and thus, the significant influence on those who have power over securities. Elon Musk made global headlines in 2018 when he was charged with securities fraud, and received a $40 million penalty from the SEC for jokingly tweeting (Posting on the Social Media platform "Twitter") that he may “take [his company] Tesla "private" at $420 a share”. Since then, regulators have been diving deeper into potentially restricting freedom CEOs have on social media. In this context, it is unclear what defines securities fraud. Little is known about the effect of social media posts on stock market volatility. As a result, CEOs may be intentionally or unintentionally manipulating the stock market through social media.

In order to explore this question, we employed several machine learning based analytical techniques on both stock market trends and CEO Tweets in an attempt to discover a relationship between the two. In general, the models predict a classification of positive or negative stock movement based on the sentiment analysis of Tweets made by the stock’s respective CEO, parallel on a time series. With respect to the entire dataset, the best performing model was the Gaussian Naive Bayes. The Multi-Layer Perceptron model trained on specific companies showed potentially promising results and the possibility of predictive power in our dataset, however we worry about potential overfitting due to the small size of the dataset.

Related Work

The area of market manipulation on the internet in general has been an active area of research in recent years. Consider, for example, *Market Manipulation and Suspicious Stock Recommendations on Social Media* [10], where Thomas Renault found that fraudsters use social media to temporarily inflate the price of small capitalization stocks. His research can be further applied to investigate if CEO activity has a similar effect, even if being fraudulent is not exactly intentional. Additionally, *Sentiment analysis on social media for stock movement prediction* [11] describes how stock price is indeed heavily correlated to sentiment, especially if the stock is particularly popular among retail investors, which demonstrates the power CEOs have in using online platforms when controlling the sentiment of the posts they make. Finally, *Stock-Price Manipulation* [12] proves how those with insider information can release false information for a profit, at the cost of other investors, especially those trading derivative securities (such as options) with weighting towards the stock. This highlights the ability and personal motivations for CEOs to intentionally use their online platforms to release false information, especially when they can simply label them as a joke (e.g, Elon Musk’s “taking Tesla private” tweet).

Dataset

Twitter Data Collection

Historical Twitter Data is a hot commodity due to its applications. Unfortunately, Twitter places a large price tag on historical data and makes it very difficult to collect otherwise, including placing restrictions on their own Developer API. After a lengthy set of difficulties in getting approved for a developer license, we attempted different methods of scraping. We leveraged some existing technologies together into a Python script that was able to collect some historical data. We executed the script on each element of an array of CEO handles, organizing scraped tweets from the web response into an array of Python dictionaries, and created CSV files named after their respective Twitter handles. We then created Python dictionaries that mapped these handles to their respective CEO Name, Company, and Stock Market

Ticker, to allow for more effective data processing. Some key attributes that were collected included tweet creation date and time, time zone, tweet, language, hashtags, handle and name, link, photo URLs, likes, number of replies, and number of retweets. We omitted useless attributes such as the link and database IDs. Due to this alternative approach in collection, there were some irregularities in our data such as missed tweets. In addition, manually scraping tweets was a cumbersome task and the project timeframe allowed us to collect data for only a limited number of users.

Stock Data Collection

The stock data was collected from Yahoo Finance, spanning the time period of January 1st, 2019, to February 28th, 2021, covering just over two years. Data was only collected for companies whose CEOs or other high-ranking employees were active on twitter, determined by previously collected Twitter data. On Yahoo Finance, downloading the stock data over a given period of time provides you with different attributes. These attributes are the date, open price, high and low price, closing price, adjusted closing price, and the volume of shares traded all on a given day. The adjusted closing price is a much more accurate determination of the actual value of a stock at close than the regular close, which corresponds to how much a stock sells for in cash at the end of the day, and is used more frequently in stock price analysis.

Since not all stocks are represented using the same currency, and the price amount a stock moves in a day is strongly related to a current stock's value, the daily percent change for each stock was calculated by using the current day and the previous day's adjusted closing price. This way, each stock could be accurately compared to each other and used in the same model, solving the problem of different stock values and currencies. In order to make the data more usable for future model building, the company name was also added to each observation in a given file. After this was done, all of the stock files were combined into a larger .csv file for future model creation.

Data Processing

Since data was collected specifically for this project, data processing was a relatively straightforward procedure. The basic processing steps are as follows: map twitter usernames to companies and tickers, select columns to keep, drop null values, iteratively process user tweets (Map tweet to associated company, get stock data corresponding to current and next date, add associated stock values (open, high, low, volume, percent change, map percent change to discrete value ('positive', 'negative', 'neutral'))

A dictionary mapping twitter usernames to companies was used to isolate corresponding stock data, where stock values were mapped to the twitter data as additional columns. The final dataset consisted of 13,006 data points. If a tweet was made on a day for which no stock data was available, we searched for the next available day, up to a maximum of five days, and mapped the next available day to the tweet instead. Similarly, tweets can be made after the stock market closes at 4PM. In the case a tweet is at a later time, we use the stock data from the next available day. Lastly, in order to account for the fact it may take some time for a tweet to take effect we map both one-to-one and next day percentage changes.

Basic Statistical Analysis

Some basic data exploration on different stocks' adjusted closing price was done in R, which required manual cross-checking of our created files. To begin, the mean and median of the percent changes on each day for all stocks combined were calculated to be 0.267% and 0.1678% respectively. While these numbers are fairly close to 0, this still implies that on average, stocks are increasing in value each day. Over the whole time period, the mean total percent change for each individual stock was 74%. Since inflation from 2019 to 2021 was only 2.88%, we can determine that the stock prices in the analyzed dataset on average went up by a very large amount. The standard deviation of the percent changes was found to be 3.169%, indicating that a stock can fluctuate quite a bit and still remain in normal range. A histogram of each stock's daily percent change was plotted, and it is clear visually that the data is

normally distributed, meaning the stocks behave in a relatively uniform fashion as a whole. There are a few outliers, but these were left in as they may have proved very important in determining which tweets corresponded to high stock movement.

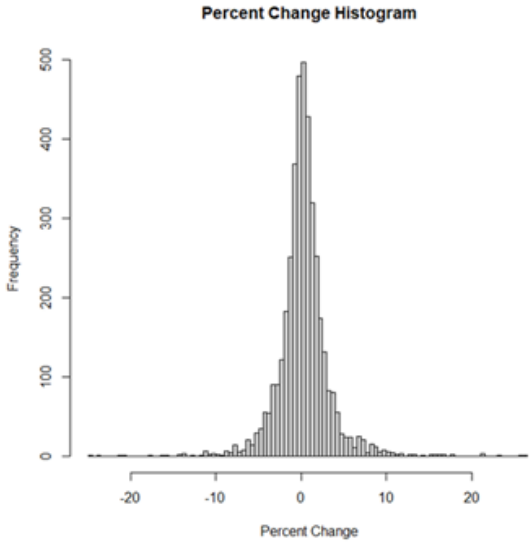

Figure 1. Distribution of percent change

The maximum positive percent change was found to be 40.7844%, while the maximum negative percent change was found to be 24.8938%. The maximum positive percent change was from Zoom Video Communications, occurring on September 1st, 2020. On this day, the CEO of Yelp, Eric Yuan, tweeted: "Earlier today we had our 2nd quarter earnings call. Thank you to all our @Zoom_us employees for your hard work. Thank you to all user's great support and trust!". On top of the related tweet having a very positive sentiment, we can see that the associated date corresponds to around the time most schools come back from summer break. Due to the COVID-19 pandemic, many lectures or other classes were being taught through zoom, corresponding to a large increase in users and attention, compounding this dramatic increase in stock price. The maximum negative percent change was from Yelp, occurring on March 16. 2020. No significant tweets were made from the CEO of Yelp, Jeremy Stoppelman, on this day.

Methodology

Sentiment Analysis

In simple terms, sentiment analysis can be defined as "the process of determining the emotional tone behind a series of words, used to gain an understanding of the attitudes, opinions and emotions expressed within an online mention"[1]. In our use case, sentiment analysis was used as a feature engineering technique in order to extract additional meaning and potential predictive power from tweets.

Before the data could be fed into the model some minor preprocessing was done which mainly consisted of removing URLs, usernames, repeated characters, stop words, punctuation, and uppercase characters. Pre-processing steps help convert noise from high dimensional features to the low dimensional space to obtain as much accurate information as possible from the text[2]. Sentiment analysis itself was done using VADER (Valence Aware Dictionary and Sentiment Reasoner), a parsimonious rule-based model for sentiment analysis of social media text, which was implemented in NLTK (The Natural Language Toolkit), a Python package for natural language processing.

VADER works by using a combination of qualitative and quantitative methods to construct and empirically validate a gold-standard list of lexical features. These lexical features are then combined with consideration for five general rules that embody grammatical and syntactical conventions for expressing and emphasizing sentiment intensity, allowing us to measure both the polarity and the intensity of sentiments[3]. Columns corresponding to the negative, neutral, positive, and compound sentiment scores of corresponding tweets were added to the dataset. As a final step, an additional column was added in which the sentiment scores were binned into discrete categories: 'positive', 'negative', and 'neutral'.

Baseline Model

The baseline model for prediction was created using a random number generator in Python 3. In this model, the odds of a stock's price going up or down was assumed to be 50%, while the chance of a stock's price having a neutral change was calculated using its frequency in the whole dataset to be 1 in 81. Using two random number generators to simulate these odds, this baseline model predicted percent change with 48.8% accuracy on average, fluctuating between 47.5% and 49.5%.

Regression

To set up the data for regression, sentence embedding was performed on all tweets in our dataset that had been previously mapped to daily percent changes. Sentence embedding was done in Python 3 using a tool called Sentence-BERT. Sentence-BERT is a known model for embedding, transferring entire sentences of varying lengths into vectors of a fixed length, making it ideal for models such as multiple regression, where the number of attributes must be equal for each observation. Sentence-BERT is primarily used for sentence comparison, because the model embeds sentences in such a way that it is very easy to determine how alike each sentence is to each other[6]. Clearly, this is very useful in regression (and model creation as a whole), as sentences with similar properties should evaluate to the same discrete percent change.

R was used to create the regression models and predict percent change for both the current day and the next day. After some data manipulation, these two models were created, and they both resulted in an average accuracy of about 52%, with observed accuracies being anywhere from 50.5% to 53.5%. Compared to the baseline model, these regression models performed 6.56% better on average.

Simple Models

Before attempting to build any deep learning models, some simple machine learning algorithms were tested. More specifically, Naive Bayes, Linear SVC, Logistic Regression, Random Forest, Adaboost, and KNN models were built and evaluated using Scikit Learn, a Python library for developing machine learning models. The basic model building steps are as follows: perform one hot encoding, drop the 'neutral' class, split the training and testing data., build a pipeline by sequentially obtaining word embeddings and a final estimator, train and evaluate the final estimator.

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation (i.e vector representations). For each ML model, a pipeline was built in which word embeddings were obtained using the TF-IDF (Term Frequency Inverse Document Frequency) Vectorizer and the model was fit to the data. The code, including the individual parameters for each model, can be found in the replication package of this report.

Bidirectional RNN

The first deep learning model built and evaluated was a Bidirectional RNN. The model was implemented using Keras, a deep learning library for Python. The basic model building steps are as follows: perform one hot encoding, drop the 'neutral' class, split the training and testing data, isolate processed tweets/sentiment category and compound sentiment scores, use pretrained GloVe model to obtain vector representations for words, build model, train for 5 epochs and evaluate model.

Word embeddings were obtained using GloVe, "an unsupervised learning algorithm for obtaining vector representations for words in which training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase linear substructures of the word vector space"[5]. In particular the *glove.twitter.27B.100d.txt* file was used. The model consists of two inputs: one takes word embeddings and passes them through a bidirectional GRU and the other takes compound sentiment scores and passes them through two fully connected layers to reduce dimensionality. The two inputs are concatenated and passed through a convolutional layer, after which average and max pooling are separately performed and merged. Finally, the output is passed through another fully connected layer in order to obtain the final predictions. The model can be found in the replication package.

```
input1 = Input(shape=(maxlen,))
x = embedding_layer(input1)
x = Bidirectional(GRU(128, return_sequences=True, dropout=0.1, recurrent_dropout=0.1))(x)

input2 = Input(shape=(100,1,))
dense_1 = Dense(128, activation='sigmoid')(input2)
dense_2 = Dense(64, activation='sigmoid')(dense_1)
x = Concatenate()([x, dense_2])

x = Conv1D(64, kernel_size=3, padding="valid", kernel_initializer="glorot_uniform")(x)
avg_pool = GlobalAveragePooling1D()(x)
max_pool = GlobalMaxPooling1D()(x)
x = concatenate([avg_pool, max_pool])
preds = Dense(2, activation="sigmoid")(x)
model = Model(inputs=[input1, input2], outputs = preds)
```

Figure 2. Bidirectional GRU

Multilayer Perceptron

The second and third deep learning models implemented were based on a multilayer perceptron (MLP) architecture, but were trained on variations of the dataset. The better performing model was trained on company-specific data: only users related to the company and the company's stock data. In an attempt to train on more data and achieve a more reasonable accuracy, another model was trained and tested on all the data, a complete collection of all companies and employee tweets. The generalized model did much worse than the company-specific model, highlighting the importance of data trained on.

The architecture was built using the Keras framework in Python and consists of 5 total layers, evident in Figure 3. The data-preprocessing was somewhat different for the MLP than the rest of the models. Rather than training on one instance of a tweet to predict the following day's stock change, the MLP trains on a day's worth of Twitter data. This process involves concatenating all influential Twitter data (tweets, likes, retweets, etc.) for a single day, merging it into one datapoint, converting all non-numerical data into numerical, and accounting for the following day's stock change for this newly merged data sample's. The few neutral cases were converted into positive cases because the neutral data showed that a positive sentiment was more common than negative, thus the conversion from neutral to positive. The tweets had to be converted into a numerical value as well, therefore a manual method to convert string values into integer values was created. The method evaluates the unicode representations of the characters in the tweets, adds the values up, and divides the summed value by the number of characters. This calculated number is then used to represent the tweets for the day. The model intakes an input vector consisting of 12 features, utilizes an Adam optimizer to efficiently learn and a dropout layer to avoid overfitting. It trains for 10 epochs, with a batch size of 128, using a binary cross-entropy loss function to determine the following day's stock change.

```
# Creating, compiling, and fitting the model to predict values
model = Sequential([
    Dense(64),
    Dense(784, activation='relu'),
    Dropout(0.33),
    Dense(12, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

model.fit(x_train, y_train, batch_size=128, epochs=10, validation_data=(x_test, y_test), verbose=2)
```

Figure 3. Multilayer Perceptron Architecture

Experiments/Results

Simple Models

The ML models were tested using Sklearn's *predict* function on the testing data from the train/test split. An accuracy score was obtained by comparing the actual labels to the predicted labels, and a confusion matrix was extracted from the predictions. The experimental results of the basic ML models are outlined below.

LINEAR SVC	Test accuracy is 0.550 (6.2% improval over baseline)
LOGISTIC REGRESSION	Test accuracy is 0.553 (6.5% improval over baseline)
RANDOM FOREST	Test accuracy is 0.542 (5.4% improval over baseline)
ADABOOST	Test accuracy is 0.547 (5.9% improval over baseline)
KNN	Test accuracy is 0.508 (2% improval over baseline)
NAIVE BAYES	Test accuracy is 0.565 (7.8% improval over baseline)

Figure 4. Simple Model performance

The best performing model was the Naive Bayesian classifier, which is based on Bayes' theorem with class conditional independence, meaning that the effect of the value of a predictor on a given class is assumed to be independent of the values of other predictors[4]. Overall, each of the ML models outperformed the baseline model by at least 2%, indicating there may be some predictive power in the tweets, but not much.

Bidirectional RNN

To train the Bidirectional RNN, the binary cross entropy loss function was used since the prediction task consisted of two categories (percent change positive or negative), and *RMSprop* was experimentally found to be the best performing optimization function. The model was trained on 5 epochs, since any more generally resulted in an increasing loss score, and a batch size of 128. The model was tested using the Keras *evaluate* function on the test set from the train/test split. Overall, the Bidirectional RNN obtained a test score of 0.744 and a test accuracy of 0.5532, which was a 6.52% improval over the baseline. Despite much trial and tweaking, the Naive Bayes model consistently outperformed the RNN.

Multilayer Perceptron

The MLP trained and tested using a binary cross-entropy loss function to predict either of the two output classes: positive (1) or negative (0). The Adam optimizer did well with the MLP model, and the accuracy appeared to reach peak results around 5-10 epochs, thus 10 epochs with a batch size of 128 was used to train. The architecture tests against the two modified versions of the dataset (company-specific and general), trained on 70% of the dataset and tested on 30%. The company-specific model achieves 100% accuracy on the majority of test cases and companies, on validation data never seen during training. A dropout layer was used to prevent overfitting but it appears to still be prevalent, otherwise the model does suspiciously well; more data is likely needed. In the attempt to generalize the model with all data, the model predicts a negative change (0) for everything, achieving a 43% accuracy. The company-specific model on average improved from the baseline by 51.2% while the generalized model decreased by 5.8%.

Threats to Validity

There exist a number of threats to the validity of these results. Firstly, the simple ML models were trained on only textual data. Incorporating the results of sentiment analysis may have improved model performance. Additionally, the TF-IDF Vectorizer was used without testing more sophisticated methods of obtaining word embeddings, such as BERT and GloVe. An additional threat to validity for all the models was the size of the dataset. The model was trained on a limited dataset of 13,006 tweets, some of which were made by the same person on the same day. Deep learning is known to often require a large, diverse dataset for optimal results. The MLP was especially threatened by a decreased sized dataset, as it took subsets of the data by merging all same-day information, and trying to use company-specific data only to train on. Thus, the company-specific model has a reasonably high likelihood of being overfit. Additionally, the discrete sentiment categories were very biased. Of all the tweets, a mere 10% were identified as having a negative sentiment. This is difficult to avoid, since it is unlikely company executives want to come off negatively, but makes this feature particularly hard to work with.

Conclusion/Future Work

The best candidates for prediction were the Multilayer perceptron, Bi-Directional Recurrent Neural Network, and Naive Bayes models. While the MLP produces high accuracy with respect to the collected data, further investigation and a larger amount of data would be needed to conclude if the MLP model is consistently accurate. Specifically, the MLP model appears to be effective for short-term prediction, implying that it may be useful in “Day-Trading”, a method of economic gain where fast intraday buying and selling takes advantage of volatility, should the model maintain its accuracy on a larger dataset. The Deep Learning based Bi-Directional Recurrent Neural Network has a lower accuracy, but appears more reliable, and may have greater performance given a larger dataset. Finally, though the Naive Bayes model has a lower accuracy for the specific application than the other two models, given its generally positive accuracy, it provides insight to the claim that CEOs may have some short-term influence on their stock depending on the sentiment of their tweets. Future work in this field may result in a more confident claim that there exists a correlation between tweet sentiments and stock movement, where more concrete and reliable models could be applied. However, stock market prediction is a historically difficult task, and simply analyzing tweets of CEOs may not prove to be enough information regarding stock prediction. In addition to the use of more complex statistical methods such as price and time series analysis, including other observations such as news articles and posts from other social media platforms could result in models with higher prediction accuracies.

References

[1] Bannister, Kristian. “Sentiment Analysis: How Does It Work? Why Should We Use It?” Brandwatch, February 26, 2018. https://www.brandwatch.com/blog/understanding-sentiment-analysis/.

[2] Jones, Anna Bianca. “Sentiment Analysis of Reviews: Text Pre-Processing.” Medium. Medium, March 15, 2018.

[3] Hutto, C.J. & Gilbert, Eric. (2015). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014.

[4] Naive Bayesian. Accessed April 15, 2021. https://www.saedsayad.com/naive_bayesian.htm.

[5] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. Accessed April 15, 2021. https://nlp.stanford.edu/projects/glove/.

[6] “Sentence Transformers Documentation.” SBERT.net. Accessed April 15, 2021. https://www.sbert.net/.

[7] Reimers, Nils & Gurevych, Iryna. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. 3973-3983. 10.18653/v1/D19-1410.

[8] Swagel, Phillip. The cost of the financial crisis: The impact of the September 2008 economic collapse. Pew Charitable Trusts, 2010.

[9] Misra, Vedant, Marco Lagi, and Yaneer Bar-Yam. "Evidence of market manipulation in the financial crisis." arXiv preprint arXiv:1112.3095 (2011).

[10] Renault, Thomas, Market Manipulation and Suspicious Stock Recommendations on Social Media (December 20, 2017).

[11] Nguyen, Thien & Shirai, Kiyoaki & Velcin, Julien. (2015). Sentiment Analysis on Social Media for Stock Movement Prediction. Expert Systems with Applications. 42. 10.1016/j.eswa.2015.07.052.

[12] Franklin Allen, Douglas Gale, Stock-Price Manipulation, The Review of Financial Studies, Volume 5, Issue 3, July 1992, Pages 503–529, https://doi.org/10.1093/rfs/5.3.503