CS230

CS 230 Final Report: Predicting US Stock Market Movement from Political Tweets

Brian Wai*
Department of Computer Science
Stanford University
brianwai@stanford.edu

Chenghao Peng
Department of Computer Science
Stanford University
cxpeng@stanford.edu

Abstract

The political environment in the US has become very unstable since Donald Trump won the election in 2016. Politics became one of the major topics on social media. President Trump's communication is largely carried by his Twitter account. The US stock market reacts to his tweet and causes volatility. With this project, we developed a few models to quantitatively and qualitatively describe how his tweets affect the US stock market. These models can be used to predict how his words on Twitter will influence the US economy. Furthermore, we investigated how some of the cutting-edge machine learning algorithms may affect the accuracy and effectiveness of these predictions.

1 Introduction

Since Donald Trump became the President of the United States, Twitter has become a significant battleground for US politics. Other politicians, Republican or Democratic, have also attempted to use twitter to reach the public regarding various issues and legislation processes. These tweets from US politicians also fed into the stock market and caused fluctuations from time to time. While people may interpret tweets differently, the aggregated interpretation of each tweet may cause the market to go up, down or stay even. We think it would be an interesting topic to investigate how the market fluctuates when new political tweets are published from these major US political accounts, as this would allow us to analyze how investors react to political events.

To predict how political tweets have an impact on the stock market, we selected intraday (1-minute interval) stock movements of S&P 500 and the tweets from the most iconic politician in Washington DC, President Trump, as the input features. We then applied different vectorizations and fed these features to different types of models, including Regular Neural Network (ANN) and Recurrent Neural Network (RNN). We tested different types of output in this project, including trends classification (stock movement) and value position at minute close (regression).

To further differentiate our project with some of the previous work, we carefully articulated the format of the input features as a combination of quantitative values (index values) and embedding features (text vectorization). More importantly, instead of trying to relate stock indices with features at the time step and making binary(up/down) predictions, this project used regression/linear layers in the models to direct predict quantitative outputs of stock movements.

*Acknowledgments: We thank Conor Smith, our project TA in CS230 at Stanford University for comments and suggestions along the way.

2 Related work

Stock market movement prediction has always been an appealing topic for researchers and investors. Early research viewed stock market price prediction as to the result of a series of random walk simulations[2]. With the increasing capacity for computing power, more and more research started to focus on using computer-based methods to solve this problem.

There have been a few past projects in CS230 at Stanford University that also focus on stock market prediction. These projects either use simple daily time-series values[8] and extract features to predict daily closing indices or they used text features to predict daily moving trends[6][5]. One big issue with daily prediction is the lack of time-sensitivity. As the stock market moves from second to second, there would not be much value even if the daily prediction is accurate.

Inspired by a few research papers that focus on studying time-sensitive data[4] and real-time predictions[7]. We think there would be more value if such models can be improved. In[3], the paper used 5-min interval intraday trading data on individual stocks and was able to prove that deep learning methods indeed have good performances with more close to real-time predictions. In[1], the authors have explored sequential models where input features are a combination of textual and numerical values.

In summary, the pursuing of close to real-time data features in the combination of the textual inputs is the main focus of this project.

3 Dataset and Features

3.1 Twitter Data

Through a Twitter Developer Account, we leveraged Twitter APIs to acquire twitter text data for selected accounts, mainly from Donald Trump's twitter account, @realDonaldTrump. The data came in .csv format with a timestamp, a type indicator and the content of the tweet.

3.2 Stock Market Data

For market data, we selected S&P 500 index as a reflection of the stock market. We picked S&P 500 index over other equity indices knowing it reflects all industries. We obtained intraday (1-minute interval) data from `https://github.com/FutureSharks/financial-data` between **11/08/2016** and **12/31/2018**. The data came with four index values: open, close, high and low. It also includes the time stamps for mapping purposes.

		Previous 10 Time Steps (mins)										
Time Stamp	Next Closing Value	0	1	2	3	4	5	6	7	8	9	Trump's Tweet
9/17/18 10:15	2898.31	2899.54	2899.9	2899.55	2899.34	2899.52	2898.79	2897.5	2897.99	2898.52	2898.7	Our Steel Industry is the talk of the World. It has been given new life and is...
9/18/18 10:11	2899.34	2899.52	2898.8	2897.5	2897.99	2898.52	2898.7	2899.1	2898.95	2899.07	2898.6	Tariffs have put the U.S. in a very strong bargaining position with Billions of...
9/18/18 10:01	2898.57	2899.32	2900.3	2899.95	2899.72	2899.16	2898.96	2898.6	2899.32	2899.56	2899.4	"A lot of small & medium size enterprises are registering very good profit...

Figure 1: Figure representing the three samples we collected over one day (09/17/2018).

3.3 Sample & Feature Selection

Between 11/08/2016 and the end of 2018, @realDonaldTrump tweeted 5920 times, within which 813 times of retweet. To create a training sample, we mapped the time stamp on the tweet to an open-market hour, which is between 9:30 am - 4 pm EST on weekdays. After mapping, there are a total of 2121 samples returned, meaning there were 2121 times @realDonaldTrump tweeted when the market was open. The 2121 samples were then split into an 80-20 train/test sets. We took the next minute closing index value as our sample label and the previous 10 closing values time-series features. Figure 1 shows a few sample examples after mapping and labeling.

Figure 2 is the plot of S&P closing values between 9:50 and 10:16 am. In this example, one can clearly see the influence of tweets coming out of @realDonaldTrump(red arrows in Figure 2), the S&P indices on the next minute of the event have clearly shifted directions. Whether or not these turns are direct market reactions to the tweets or normal market fluctuation is not clear at the moment and is what the project can help to predict moving forward.

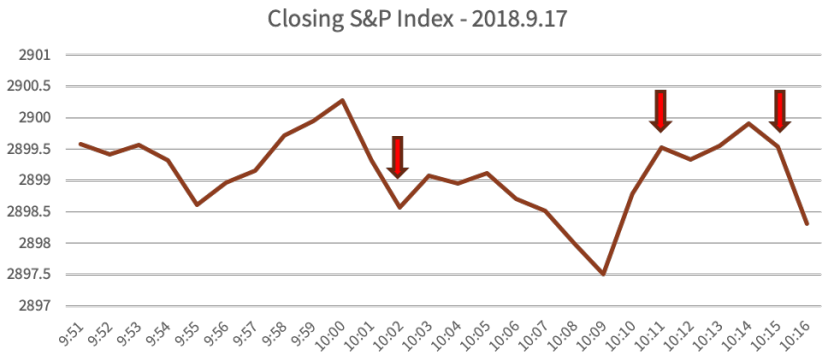

Figure 2: Figure representing how each of Trump's tweet of the day may have possibly shifted/strengthened the market's moving trend.

3.4 Pre-processing

The samples we collected above in Figure 1 need pre-processing steps for both the textual and numerical parts.

For textual features, we encoded the tweet text at the time of the event. We selected two word-encoding methods: one-hot full features (OneHot) and 200 dimension Global Vectors for Word Representation(GloVe) we acquired from `www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`. Training word2vec embedding over the tweet text collected was forced to drop due to the limited time and computing resources. There are a total of 6448 features for the one-hot encoding.

Normalization methods were applied separately both the numerical and textual values to ensure the features will present similar weights when training. We chose MaxMin scaling over Z-score normalization for the little differences between the two methods when tested.

We combined the features in various ways for the different models. For flat models (baseline and NN), the 10-step time-series is flatten and added to the textual feature N_T. The total number of features for each sample $N_{total} = 10 + N_T$. For sequential models(LSTM), features were separated into time steps. The numerical value of the time step is combined with zero vector that has same length of the text vector for the 9 steps prior to the event. At the time of the event, numerical value is added to the encoded textual vector, making the the total number of features at each time step: $N_{total,t} = 1 + N_T$. Figure 3 and 4 in the method session show how these feature manipulations are done.

We compared the stock index value at the minute after the event with the value at the minute of the event to generate labels. If the change is too subtle(less than 0.5 point or 0.015%), we consider it has no impact on the market. Otherwise, the stock market can either go up or down, making it a three-class classification problem. We also used the actual numerical values for regression models.

4 Methods

We explored three types of algorithms in this project: a baseline linear model (Baseline), a flat fully connected Neural Network (NN) with three layers and a sequential LSTM model.

4.1 Linear Model

The linear models we adopted are linear regression for regression and softmax for classifications. They are very standard linear baseline models with basic formula shown below:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_n x_{in} = x_i^T \beta \tag{1}$$

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ where: } z_i = y_i \text{ in (1)} \tag{2}$$

4.2 Fully Connected Neural Network(FCNN)

Fully connected neural networks are neural networks where neurons are connected to each other in various patterns, to allow the output of some neurons to become the input of others. In our model, we adopted 2 fully connected layers with a final output layer either as linear regression or softmax classification to serve for regression or classification problems. Figure 3 shows an illustration of how the time series data is combined with encoded textual features and fed into the baseline and FCNN models.

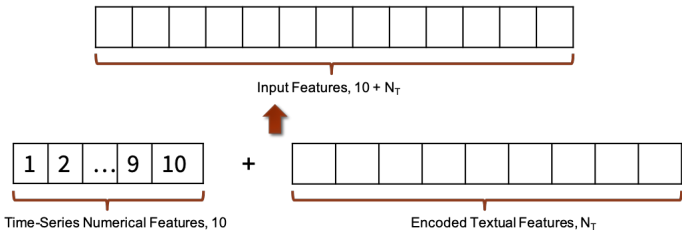

Figure 3: Figure showing how the time-series features are combined with text features for flat models.

4.3 Long short-term memory(LSTM)

Long short-term memory (LSTM) algorithm is a special type of Recurrent Neural Network(RNN). The LSTM/RNN model is a sequential model that addresses the strong correlation between inputs such as time series. LSTM further addresses the diminishing gradient problem in RNN which allows information to be carried over time. In this project, we adopted a 10 time-step LSTM model with 64 hidden units. The structure of constructing the features and architecture flow is shown in Figure 4.

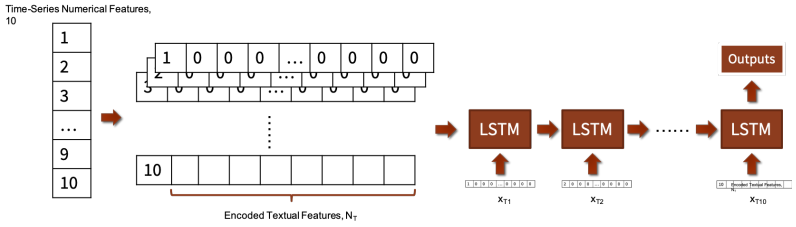

Figure 4: Figure representing how the time-series features are combined with text features for LSTM sequential models as well as how the features are fed into LSTM units to produce the final outcome.

5 Results/Discussion

5.1 Hyper-parameter selection

For this project, we experimented with different learning rates and different methods of regularization. We picked learning rate 0.01 for FCNN and logistic regression, and 0.1 for LSTM given the reasonable epochs (between 10 - 50) we have to take to reach the desired results. In the classification problem, With 0.01 the learning rate for FCNN, the test set F1 scores were stabilized with epoch over 15. Similarly for the LSTM, the test set F1 score stabilized after epoch 20. We tried different regularization and drop out rates as well. None of which resolve the overfitting issue we observed in classification. To accommodate a stateful LSTM model, we picked the batch size of 1 to train the models. We chose a 0 dropout rate because we needed to learn complex functions.

5.2 Classification

As shown in Figure 5, training set F1 scores are dashed lines and test set F1 scores are solid lines. None of the classification models turned out to have produced valuable results. Neither they have outperformed each other. We did, however, see different model behaviors as features and input

value varies. Both baseline and LSTM with one-hot encoding stabilized very fast after a few epochs, it maintains large over-fitting with no help from dropouts or regularizations. With GloVe word embedding, LSTM possesses the least amount of over-fitting and baseline with word-embedding never even stabilized. Overall, the classification models pose rather disappointing results. The test F1 scores centered around 0.45, which is merely better than random guessing for a three-class classification. (F1 Score = 0.33).

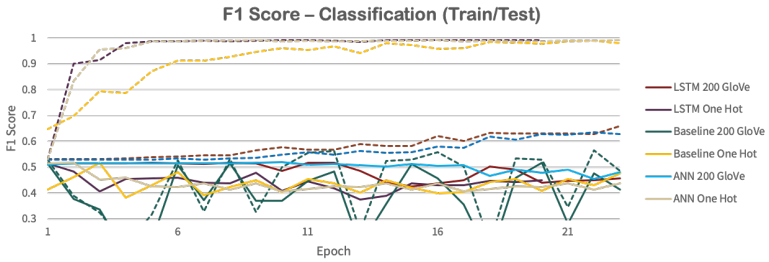

Figure 5: Figure representing different trend of model reaching stabilized F1 Score over number of epochs, training sets in dash and test sets in solid.

5.3 Regression

In contrast, quite interestingly, all models have returned decent results for regression fitting, compared to that of the classification model. With ANN with one hot encoding showing the worst fit, performance, and accuracy, the model experienced several times of exploding coefficients as well without regularization. The two best models are linear baseline with GloVe embedding and LSTM with one-hot encoding. They both have achieved very small over-fitting and high performance. As expected, one-hot encoding did not only pose the worst performance on the ANN model, it also over-fitted the baseline model regardless of the high-performance evaluation over the training set.

Model Name	Fit (R^2)	Performance (RMSE)	Accuracy (MAE)
Baseline-OneHot (train/test)	0.99/0.99	0.33/7.36	0.32/2.04
Baseline-GloVe200 (train/test)	0.99/0.99	2.94/3.81	1.08/1.26
ANN-OneHot (train/test)	0.99/0.98	87.3/423.9	7.22/15.98
ANN-GloVe200 (train/test)	0.99/0.99	90.25/100.26	7.08/7.82
LSTM-OneHot (train/test)	0.99/0.99	4.14/6.82	1.41/1.78
LSTM-GloVe200 (train/test)	0.99/0.99	27.68/21.2	2.63/2.01

Table 1: Table summaries the evaluation metrics that were used in this project for regression performance evaluation over different model and feature set.

6 Conclusion/Future Work

Predicting the stock market is very difficult. Our model usually did very well on the training set but only did little better than random guessing on the test set in classification. We tried implementing regularization to help combat this, but regularization only made performance on the test set worse. This indicates the way we label the samples may have some biases, which we need to further look into.

The primary purpose of this paper was to see what predictions can be made on a tweet without using semantic analysis, which we achieved by using regression against text encoding directly.

Future work also remains to be done in gathering other politician's tweets and evaluating results, and seeing if a model trained on one politician's tweets can generalize to others. Researchers could customize this methodology to estimate GDP given the extensive history of tweets and relative strong correlation between GDP and S&P 500. Furthermore, connecting tweets to stock market data would be a fascinating study. For example, how long after a tweet does the stock market change? The true value of this methodology is that it overcomes, to some significant degree, the bias inherent in tweet. Thus, it makes it particularly valuable for customizing to a wide range of problems.

7 Contributions

Chenghao did: writing reports, data collection and processing (including word2vec), LSTM, running cloud.

Brianwai did: editing reports, basic NN implementation and linear regression, with regularization.

8 Code Repo

`https://github.com/cxpeng9223/CS230-Project`

References

[1] Ryo Akita et al. "Deep learning for stock prediction using numerical and textual information". In: *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*. IEEE. 2016, pp. 1–6.

[2] Johan Bollen, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market". In: *Journal of computational science* 2.1 (2011), pp. 1–8.

[3] Eunsuk Chong, Chulwoo Han, and Frank C Park. "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies". In: *Expert Systems with Applications* 83 (2017), pp. 187–205.

[4] Xiao Ding et al. "Deep learning for event-driven stock prediction". In: *Twenty-fourth international joint conference on artificial intelligence*. 2015.

[5] Jason Kurohara, Joshua Chang, and Callan Hoskins. "Predicting Stock Market Movements Using Global News Headlines". In: *CS 230 Project* (2018).

[6] Cheryl Ji Jimmy Qin. "Natural Language Processing and Event-driven Stock". In: *CS 230 Project* (2018).

[7] Ruoxuan Xiong, Eric P Nichols, and Yuan Shen. "Deep learning stock volatility with google domestic trends". In: *arXiv preprint arXiv:1512.04916* (2015).

[8] Kevin Li Glenn Yu. "Deep Learning for Stock Price Forecasting". In: *CS 230 Project* (2017).