Predict Effect of President Trump's Tweets on Stock Market Movements

TONG YANG, YUXIN YANG
tongy@stanford.edu, yuxiny@stanford.edu

Introduction

Background: Big high frequency finance corporates trade on Trump's Tweets in large quantities with algorithms, which can affect the market price. We want to understand the impact and implication of President Trump's Tweets on S&P 500 Movement, and predict how the market fluctuates after their release.

Goals: Given President Trump's Tweets (each tweet as a single data sample), here are our objectives:

- Predict stock market price change (rise/remain/same/drop)
- Predict if major market move happened, or not

Problem Statement

Graph illustration for Binary Prediction at One Time Stamp

Datasets

1. President Trump's Twitter Archive from 2009 till now: http://www.trumpvwitterarchive.com/
   including 11,330 tweets with average length 20 words (6 MB)
2. Wharton Research Data Services (WRDS)
   TAQ ( Consolidated Trades) – SPY (S&P 500 ETF Trust)
   470,159,618 trade entries containing raw SPY price.
3. GloVe: Global Vectors for Word Representation
   We downloaded GloVe, a pre-trained model for word embedding. It contains matrices representations of words (400,000 words as 50, 100, 200 and 300 dimensions respectively).

1. Millisecond Resolution S&P500 Data to Minute Resolution

   In order to map the price change to the release timestamp of Trump's Twitter, we parsed our millisecond SP500 market data to minute resolution by taking the first reading of a particular minute. For minutes within which no trade happened, we applied forward fill.

2. Sentence Processing, Removal of Stop Words for Naive Bayes and SVM

   We used SKLearn NLTK library in Python for Words Pre-Processing. We transformed each word to lower case regular form, and removed punctuations and stop words with high frequency. A typical final sample is shown below (Final Input Data for Baseline Models).

   Sentiment analysis
   1. The return looks too good to be true.
   2. If you can commit, you win.

   3. Words Embedding and the Skip Gram Model for RNN

   Trump's Tweets have ~900 unique words. We downloaded pre-trained vector representation of words from GloVe, and converted each word from 9000 dimensional one hot representation to a 300 dimensional vector that captures the most information for that word to feed into RNN model.

   Models and Algorithms

   1. Naive Bayes Classifier Model

   Naive Bayes classifier only considers each word in Trump's Tweet to contribute independently to the probability that the market goes up or down.

   \[ P(y | x_1, \ldots, x_n) = \prod_{i=1}^{n} P(y | x_i) \]

   Where \( x \) shows the existence of each word, and \( y \) represents classification result (up, down, or no change). We picked this model as the simplest baseline for our purpose to predict binary change.

   2. Support Vector Machine Classifier Model with Non Linear Kernels

   SVM model is an ideally more robust model than Naive Bayes, for it can capture some inter-correlation between two words to some degree and optimize the decision boundaries. We eventually ended up using Poly Kernels observing it outperforms other kernels.

   3. Recurrent Neural Network (LSTM) Classifier

   LSTM would theoretically outperform, because it has the ability capture long term messages in a sentence. It can also remember "state" information that can capture some latent features hidden in a sentence. Our model used Softmax Cross Entropy between Logits and Labels as loss function.

Experiments and Toolsets

- We used Python SKLearn Naive Bayes, and SVM library respectively to train our baselines. After fine tuning learning rate, loss, number of iterations and so on, we found optimized learning rate for SVM classifier using stochastic gradient descent to be 0.00001 and max iteration to be 1000.

- We used Python Tensorflow Library to train our RNN LSTM model. Due to the limited size of dataset, we used batch size of 24 with 5 LSTM units for model training, and we decided not to use a stacked LSTM model to prevent over-fitting.

Results and Analysis

Naive Bayes: Accuracy on 2000 Test Data 42.5%

SVM Classifier: Accuracy on 2000 Test Data 46.5%

Recurrent Neural Network (LSTM): Our best LSTM model (30 words, 300 dimensional embedding) converged to have an final test accuracy of 48.2%.

Conclusion & Future Work

Conclusion: All three models we used turns out to outperform the baseline random prediction, and LSTM is the out winning model because it models feature indicators and also performs semantic analysis, which can better capture tweets meaning. From another point, in addition to the interval testing, it seems that it takes some response time for market price to change after tweet release.

Future Work: 1. GPU training to finer tuned model; 2. Testing the robustness of model by testing on other news and Twitter releases.