Novel Approaches to Sentiment Analysis for Stock Prediction

Chris Wang, Yilun Xu, Qingyang Wang
Stanford University
`chrwang, ylxu, iriswang @ stanford.edu`

Abstract

Stock market predictions lend themselves well to a machine learning framework due to their quantitative nature. A supervised learning model to predict stock movement direction can combine technical information and qualitative sentiment through news, encoded into fixed length real vectors. We attempt a large range of models, both to encode qualitative sentiment information into features, and to make a final up or down prediction on the direction of a particular stock given encoded news and technical features. We find that a Universal Sentence Encoder, combined with SVMs, achieves encouraging results on our data.

1 Introduction

Stock market predictions have been a pivotal and controversial subject in the field of finance. Some theorists believe in the efficient-market hypothesis, that stock prices reflect all current information, and thus think that the stock market is inherently unpredictable. Others have attempted to predict the market through fundamental analysis, technical analysis, and, more recently, machine learning.

A technique such as machine learning may lend itself well to such an application because of the fundamentally quantitative nature of the stock market. Current machine learning models have focused on technical analyses or sentiment as a single feature. But since the stock market is also heavily dependent on market sentiment and fundamental company information, which cannot be captured with a simple numeric indicator, we decided to create a machine learning model that takes in both stock financial data and news information, which we encode into a fixed-length vector. Our model tries to predict stock direction, using a variety of techniques including SVMs and neural networks. By creating a machine learning model that combines the approaches of technical analysis and fundamental analysis, we hope our model can paint a better picture of the overall market.

2 Related Work and Analysis

Sentiment analysis and machine learning for stock predictions is an active research area. Existing work to predict stock movement direction using sentiment analysis includes dictionary based correlation finding methods, and sentiment "mood" detection algorithms.

Several papers, such as Nagar and Hahsler [1], propose building a corpus of positive and negative words commonly used in financial news, and using the counts of each of these words in news headlines to get a NewsSentiment value for each input news; they then model the relationship between NewsSentiment and the stock movement. Bollen et al. [2] focuses on using Twitter sentiment and Google's Profile of Moods Algorithm to train a machine learning model to categorize snippets into one of several sentiment categories, before using the category as a feature in a Self Organizing Fuzzy neural network.

The related work showed us significant promise in using sentiment to predict stock movement. However, we were concerned that a single sentiment value was inadequate to capture the complexities of the news and company information. Thus, we decided to design a model to take in news data more robustly, namely by encoding news to real vectors, and feeing the entire vector to a classification model. Past work that encodes text to vectors uses various skip-gram algorithms [3], such as Google's Universal Sentence Encoder (Cer et al. [4]), though we could not find an existing application to financial stock predictions.

Github link: https://github.com/Beehamer/cs229stockprediction

3 Dataset and Features

3.1 Data sources

Our dataset is composed of trading, macro, technical and news data related to 20 NASDAQ companies, from 2013 to 2017. We used the Yahoo Finance API to extract trading-related information on each stock ticker, including price and volume, on a daily basis. We also extracted overarching macro-data including quarterly GDP, CPI, and daily Libor from the Fed website. In addition, we computed technical indicators including CCI, RSI and EVM from trading data. Finally, we scraped daily news headlines and snippets for each ticker from New York Times and Google News.

3.2 Data preprocessing

We used a few approaches to merge and preprocess the data. To match quarterly macro data (e.g., GDP) with other daily data, we assumed the macro features to be constant throughout the same quarter. We processed the news data using text representation and sentiment analysis models, which will be discussed in detail in section 4, before merging it to the full data set. For tickers which have multiple news for certain dates, we averaged the sentiment/ encoded vectors for Google news and used the top 1 news for New York times because New York times ranks top articles. For tickers which don't have news articles on certain dates, we replaced the missing value with the latest available news. We choose not to normalize the data to avoid destroying correlations of the sparse matrix. Furthermore, we classified the 1-day (next-day) stock movement into a binary label Y, where $Y = 1$ if adj. close price $\geq$ last adj. close price and $Y = 0$ if adj. close price $<$ last adj. close price.

Finally, we built two datasets using news from New York Times and Google, respectively, each of which contains 24K entries and 70 features. We split all samples before 2017 into the training set and hold out the rest as test set.

3.3 Data visualization

We plotted label Y on the first two principal components of news data. The plot reveals the complicated nature of the features, implying that high-dimension classifiers are required for the dataset.

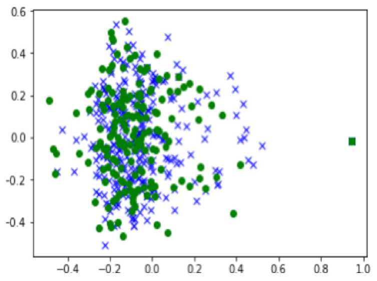

Figure 1: Label vs. two principal components of news

4 Methods

4.1 Model overview

In general, the problem is a supervised-learning problem, i.e., we are predicting the next-day movement of the stock by taking in the trading information about the stock, and the information from the ticker-specific daily news. The task can be split into two parts, namely, to represent the news as a fixed-length real scalar or vector, and to use the news, together with trading information, technical indicators, and macro data, to make the prediction. In order to capture the semantic information of the text and represent it in a vector space, we eventually decided to use a Google Universal Sentence Encode (USE) as the encoder (section 4.2). In terms of the stock prediction model, which is trained to take in all of the technical and encoded features to make the prediction, we used Logistic Regression, Random Forest, SVM, and a variety of neural networks (section 4.3).

4.2 Text representation

The text representation model is required to convert news sentences and snippets to real-value fixed-length scalar or vector representations that can be used in the stock movement prediction model. The goal is to have the model best capture fine-grained semantic and syntactic information.

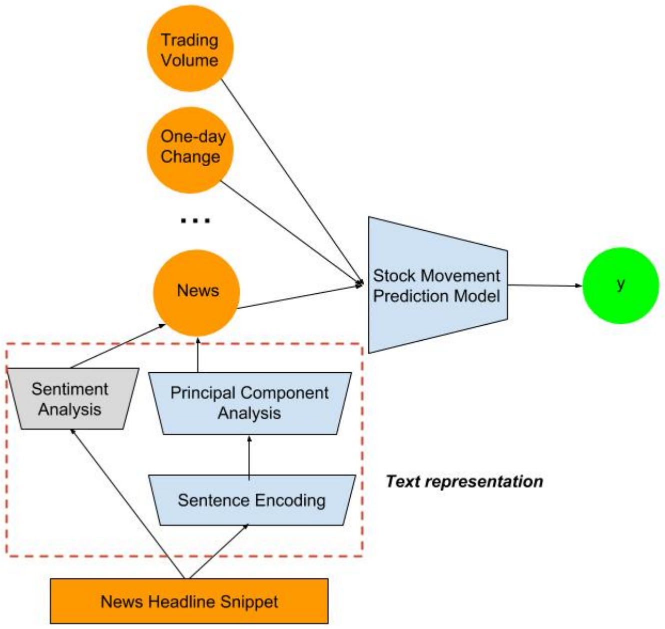

Figure 2: The general model structure

One method we tried was to directly extract a sentiment signal scalar from a given sentence vector. Dictionary based approaches use statistical information of word occurence count to extract useful information. We used a "SentimentAnalysis" package from R with a financial dictionary, similar to the one mentioned in related work, to get a scalar sentiment score for each of our sentences. We also tried using a pre-trained sentiment LSTM model (which was trained using labeled data from CNN News [5]) to extract the sentiment from the headline and snippet text. However, neither of the methods mentioned above achieved a reasonable accuracy in making the overall prediction, and a plausible reason is that the high-level sentiment information is not sufficient in representing the text.

Thus, we used sentence embeddings to produce a vector space with a more meaningful substructure to represent the news, and fed the entire vector embedding into our classification model. Recent methods of finding fixed length real vector representations of words and sentences have succeeded in a wide range of tasks from sentiment analysis to question and answering. These models can be broadly divided into word encoding methods and sentence encoding methods. To evaluate each of these models to choose one for us to use, we took several sentences, and compared the results of the encoding to see if the encoders captured the similarities and differences between sentences.

Word encoding strategies include Word2Vec, ELMo, GloVe, and FastText. These models use the bag of words technique, which detect how often words appear in similar context of other words to get a vector representation of each word (though the FastText actually goes character by character). We noticed that one problem with using one of these word encoding strategies on our sentences is that it does not consider the words of the sentence together, and we are unsure about how to composite the words to the sentence.

Thus, we decided to choose a method that encoded entire sentences such as Skip-Thoughts (similar to Word2Vec but with sentences instead), InferSent (looks at pairs of sentences), or a Universal Sentence Encoder. The USE consists of a Deep Average Network (DAN) structure–although this structure also takes an average of words, there are layers of dropout that allow important words to be highlighted. There was also another variant of the USE that used a transformer module, a novel neural network architecture based on a self-attention mechanism of context; this method achieves the best in detecting sentence similarities, however, we found this technique to be too slow on our data. Eventually, we decided to use the pre-trained Google DAN USE as our sentence representation because of its ability to detect features in a large range of sentence types, including our news, large pretrained corpus, and dropout technique. [4]

We also use Principal Component Analysis, which projects the data to the dimensions where it has most of its variance, to reduce the dimensions of the output of the DAN from 512 to 20, in order to enhance the computational efficiency.

The PCA is based off of all of the seen vectors in the training set, and the principal components stay the same for the test set.

4.3 Stock movement prediction

Logistic Regression is used as a baseline to map the features to stock movement.

Random Forest which constructs a multitude of decision trees at training time and outputs the class that is the majority vote of the individual leaves, is widely used for classification and regression. We tuned depth of the tree and leaf size to regularize the model.

Support Vector Machines are mentioned in previous research [6] to be effective in stock prediction applications. The RBF kernel captures the high-dimensional nature of stock movement. We can regulate the model by using different costs C. $L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y^{(i)}(\Theta^T\phi(x) + b)) + C||\Theta||_2^2$

Fully-Connected Neural Networks are composed by interconnected neurons whose activation functions capture the non-linear relationship between the variables and the binary result. We tuned the model on different parameters and the best-performance model structure consists of two hidden layers (50/2 or 10/10) with ReLU activation and a learning rate of 1e-3, although it did vary based on dataset.

Convolutional Neural Networks have been widely used for image processing. We thought it might be effective to do convolutions over the sentence embeddings because of their structure; however, we also acknowledge that because of the PCA and the way the Google USE DAN works, the adjacent features may not be relevant to each other. Two 1D-conv layers, each followed by a pooling layer, are included before the final fully connected layer. We picked the learning rate with which the model converges most effectively–1e-3.

Recurrent Neural Networks are proven to be effective in dealing with sequential data, with the output being dependent on the previous computations [7]. The hidden layer captures information about what has been calculated so far. x_t is the input at time step t, which is the feature variable mentioned above. s_t is the hidden state at time step t, the memory of the network. s_t is calculated based on the previous hidden state and the input at the current step: $s_t = f(Ux_t + Ws_{t-1})$. We are training separate RNNs for each ticker. The learning rate is 1e-4.

5 Experiments/Result/Discussion

5.1 Experiments and results

To examine the model stability, we trained each model on the two datasets using NY Times and Google news separately with a learning rate mentioned in 4.3 respectively. The mini-batch size selected was the largest possible value to fit into CPU; for RNN the mini-batch is all data for each ticker. We evaluate the model using mainly test accuracy. Meanwhile, we also monitor the f-1 score to ensure balanced performance on both 0 and 1 labels. As shown in table 1, SVM with RBF kernel is the best performing model on both datasets. Neural network and CNN also achieved decent performance. However, results from logistic regression and random forest are not satisfactory.

Table 1: Experiments and results

	Google News		NY Times	
	Training acc.	Test acc.	Training acc.	Test acc.
LR w/o news	0.5280	0.5169	0.5280	0.5169
LR w/ news	0.5337	0.5046	0.5308	0.5185
Random Forest	0.7770	0.4870	0.7601	0.5006
SVM (RBF)	0.5650	0.5430	0.6005	0.5414
NN	0.6172	0.5273	0.5881	0.5259
CNN	0.5816	0.5100	0.5464	0.5204
RNN	0.4931	0.4809	0.4832	0.4695

5.2 Discussion

Best model: **SVM with RBF** kernel is able to project the features onto a high-dimensional space and effectively separate the labels. We tuned the cost parameter to prevent overfitting. Precision and recall rates of the best performing models on Google News and NY Times are shown in table 2. Although we attempted to achieve a balanced performance on 0 and 1 labels, the selected model still outputs a relatively imbalanced confusion matrix. We believe that such issue is raised by our loss function, which is designed to maximize the overall accuracy but not to ensure the performance on both labels.

Table 2: Precision/recall of SVM on test set

	Google News		NY Times	
	Precision	Recall	Precision	Recall
$Y = 0$	0.48	0.24	0.47	0.33
$Y = 1$	0.56	0.78	0.57	0.71

Bias: Data visualization reveals that our dataset is not separable in a low-dimension space, which explains why **random forest** and **logistic regression**, with simple structure, are not working well.

Variance: Random forest shows the overfitting problem even after regularization. Our dataset contains features which might be positively or negatively correlated with each other, e.g., vectors representing news headlines. Selecting a subset of such features may not be able to reduce the variance efficiently.

Stability: As mentioned before, for the **RNN**, in order to capture the time-series nature of each stock, we split the dataset by ticker before running the model, which in turn shrunk the data size. Additionally, some of the tickers had relatively sparse unique news data. Furthermore, it is probable that the deep structure of the model caused the gradient update to be inefficient. We also found that the RNN is very sensitive to the initialization of the hidden state, which shed some light on the inefficacy of back-propagation. To fix this, we might change the structure of hidden state or use a different activation function. These are some possible reasons the RNN outputs a high proportion of 1s or 0s on some of the subsets and cannot be used as a stable model for future predictions.

To gain better understanding of the model performance, we plotted the true and predicted stock movement of Facebook in 2017 as follows, where the same color on the same day indicates correct predictions. Examining the predictions closely, we found that the best performing model (SVM) is more able to detect major up / downs than smaller changes.

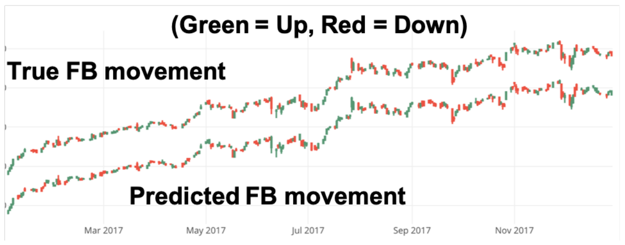

Figure 3: Prediction of Facebook stock movement in 2017

6 Conclusion/Future Work

In conclusion, we think stock-specific news might help in predicting next-day stock movement. However, it is hard to turn such informational edge into a profitable trading strategy given that we are merely predicting ups and downs. In addition, our model seems to be more able to detect major movements than smaller ones. We believe the following steps can be taken to improve model performance in the future:

- **Customized loss function**: We think achieving high accuracy and balanced performance on 1 and 0 labels are both important in stock movement prediction. However, the second goal was not built into the loss function of our models. As the next step, we can customize the loss function (e.g., as binary cross-entropy) to obtain a more balanced performance.
- **Enhance data quality**: To make the project usable in real life, we built the dataset using news we scraped from the internet. Such data might include irrelevant or inaccurate news which increases noise. In the future, we think adding more cleaning techniques and including models to detect unhelpful news may help.

7 Contributions

Our team spent 50 percent of our time on collecting and preprocessing data, 20 percent on text representation and 30 percent price movement modelling and debugging. Given the challenging nature of our topic, three of us worked closely during the whole process. Chris contributed primarily to collecting the trading data, working on sentiment signal modelling using text representations, and applying the models to New York Times data. Yilun contributed primarily to collecting sentiment data, and testing and debugging the RNN and CNN models. Iris contributed primarily to collecting sentiment and trading data, data preprocessing, and applying the models to Google News data.

We would like to thank the entire CS 229 teaching staff, including our mentor Atharva Parulekar, for providing invaluable feedback thorughout the course of the project.

8 References/Bibliography

[1] Nagar, A. & Hashler, M. (2012) Using Text and Data Mining Techniques to extract Stock Market Sentiment from Live News Streams

[2] Bollen et al. (2010) Twitter Mood Predicts the Stock Market. *Journal of Computational Science*

[3] Heidenrich, H. (2018) Paper Summary: Evaluation of sentence embeddings in downstream and linguistic probing tasks

[4] Cer, Daniel, et al. (2018) Universal Sentence Encoder. *Google Research*

[5] https://github.com/clairett/pytorch-sentiment-classification/

[6] Madge, S., Bhatt, S. (2015). Predicting Stock Price Direction using Support Vector Machines. *Independent Work Report Spring*

[7] Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., Soman, K. P. (2017, September). Stock price prediction using LSTM, RNN and CNN-sliding window model. In Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on (pp. 1643-1647). IEEE.

9 Github repository

https://github.com/Beehamer/cs229stockprediction