CS230

Predicting Stock Trends From News Articles

Sahil R. Nayyar
Department of Electrical Engineering
Stanford University
srnayyar@stanford.edu

Abstract

This paper details the background, development, and results of a deep learning model designed to solve the problem of predicting stock trends from news articles. This task is framed as a classification problem: given a temporal sequence of news articles, the goal is to predict whether a stock's price will increase, decrease, or stay around the same. To solve this task, I aimed to use a recurrent neural network with trainable attention parameters. Although I did not get to this point, I was able to get reasonable performance with a dense neural network.

1 Introduction

Forecasting the stock market is not easy. There are a vast number of factors that influence stock trading behavior, and stock prices are especially volatile to short-term fluctuations. Traditional approaches to this task involve a combination of technical analysis, which focuses on extracting insight from historical trends (e.g. using past prices to predict future prices), and fundamental analysis, which employs a bottom-up study of a company's financial statements, press releases, and other contextual information to get a sense of the "bigger picture."

More recently, the rise of effective deep learning has prompted research into automating stock forecasting. Although efforts to augment technical analysis have been largely unsuccessful, several studies have demonstrated the efficacy of incorporating deep learning into fundamental analyses; their relative success is probably due to an emphasis on the stock market's underlying forces.

My project aimed to develop a model that, given a fixed-length sequence of news data relevant to a certain public corporation, predicts whether that stock's price will increase, decrease, or stay around the same. This approach treats the task as a "trinary" classification problem: instead of trying to predict the stock's price directly, it defines a pair of thresholds in which to place the stock's rise percent, which is given by the following formula:

$$\text{RisePercent}(t) = \frac{\text{OpenPrice}(t+1) - \text{OpenPrice}(t)}{\text{OpenPrice}(t)}$$

This paper presents the development of such a model with a temporally and news-level weighted recurrent neural network. Unfortunately, the development itself was not completely successful, yet there are still some insights that can be taken away from this project.

2 Related work

This project, in both its problem of interest and architecture, primarily draws inspiration from a study published by Hu et al., 2018. [2] Here, they make use of five layers: a *news embedding layer*, in which the sequence of news articles is batched into daily corpora and embedded through a pretrained encoder, a *news-level attention layer*, in which the encoded corpora are averaged together with trained weights, *a bidirectional recurrent layer*, a *temporal attention layer*, in which the entire sequence is averaged together with trained weights, and a *dense trinary softmax output*. This model stuck out to me the most due to its resemblance to the process of human learning, namely due to its ability to prioritize different news articles and time periods. With their model, the authors were able to get an accuracy of around 0.48.

Some researchers have used models less complex than neural networks and have gotten successful results: for example, Zhou et al. (2016) used a support vector machine to predict discretized (via 5-means) stock prices from "emotion" features inferred from Weibo tweets [5], and Si et al. (2013) employed a probabilistic graphical model (specifically, a Direchlet Process Mixture model) to predict binary stock trend outputs (i.e. “UP" or “DOWN") from topic-based Twitter sentiments [3], both of them acheiving accuracies of 0.58 and 0.68, respectively. There have also been several successful studies that employed both fundamental and technical analyses; for example, Akita et al. (2016) combined both news articles and daily open prices to predict closing prices using a Long Short-Term Memory (LSTM) network, and the resulting model was able to generate a profit for them [1]. Wang and Gupta (2013) tried to use a denoising technique called wavelet analysis to aid in future stock price prediction; however, they found that doing this removed too many fine details, and made their model perform worse than without [4].

3 Dataset and Features

To obtain the news data, I built a web scraper in Python to download 593,997 articles from www.reuters.com from dates ranging from /2016 to 2018. For the stocks data, I downloaded closing prices for the top 20 Nasdaq companies from https://www.alphavantage.co/'s RESTful API endpoint. A major issue with the data collected is that, for many of the smaller companies, there are many days in which no articles appear (Figure 1). All articles for which there were no references to any of the 20 companies were discarded.

In order to integrate the collected data into the models' data pipeline, I made use of a pretrained Word2Vec embedding created by Google, Inc. It was trained on a corpus of Google News articles, and maps a vocabulary of around 3 billion words to a space of 300-dimension English word vectors. For each news article, I generated its encoding by first using the Word2Vec model to compute a vector for each of its words and then averaging these vectors to obtain a vector representative of the overall article. All words were stripped of any non-alphanumeric numbers before being fed into the Word2Vec model, and if a word was not present in the Word2Vec's vocabulary, a randomly-chosen word's vector was chosen for it.

The original dataset of 583 dates (2017-05-01 to 2018-12-04) was partitioned into a training set spanning 488 dates (2017-05-01 to 2018-08-31) and a dev set of 61 dates (2018-10-01 to 2018-11-30).

4 Methods

This section describes the architecture of the final model implemented for this project, which was iteratively built up to by adding layers step-by-step – this will be further discussed in the next section.

First comes the news-level attention layer. We begin with the fixed-length sequence of news corpora $\{n_1, n_2, \ldots, n_{t-1}, n_t, n_{t+1}, \ldots, n_T\}$, where each corpus $n_t = [n_{t1}, n_{t2}, \ldots, n_{ti-1}, n_{ti}, n_{ti+1}, \ldots, n_{tL_t}]$ is a variable-length list of Word2Vec encoded news articles $n_{ti} \in \mathbb{R}^{300}$. For each corpus n_t, we aggregate the news articles n_{ti} weighted by attention values α_{ti}, that depend on learnable parameters W_n, b_n common to all articles:

$$u_{ti} = \sigma(W_n n_{ti} + b_n)$$

$$\alpha_{ti} = \frac{e^{u_{ti}}}{\sum_j e^{u_{tj}}}$$

$$d_t = \sum_i \alpha_{ti} n_{ti}$$

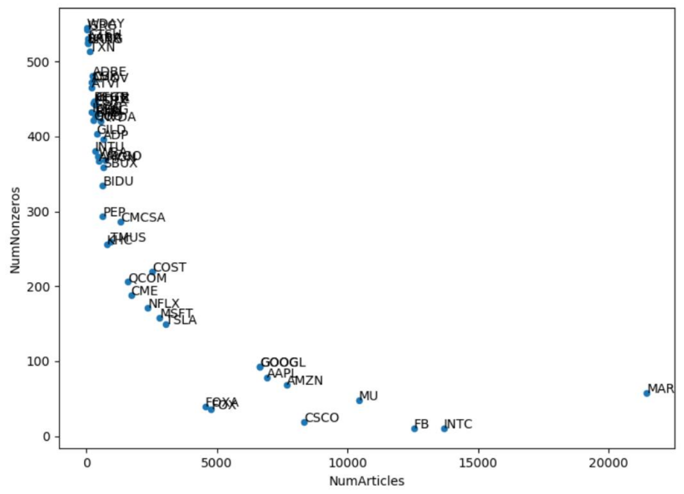

Figure 1: A visualization of the relationship between the number of articles and the number of days in which there were no articles, labeled by company.

Next comes the recurrent portion, which consists of a gated recurrent unit (GRU) that produces the sequence $\{h_1, h_2, \ldots, h_{t-1}, h_t, h_{t+1}, \ldots, h_T\}$ as its output. The current state h_t at any time t is a linear interpolation of the previous state h_t and the current update $\tilde{h}_t$:

$$\tilde{h}_t = \tanh\left(W_h d_t + r_t(U_h h_{t-1}) + b_h\right)$$
$$h_t = (1 - z_t)h_{t-1} + z_t \tilde{h}_t,$$

where r_t quantifies the previous state's importance and z_t quantifies the update's importance, both of which are sigmoidal outputs:

$$r_t = \sigma(W_r d_t + U_r h_{t-1} + b_r)$$
$$z_t = \sigma(W_z d_t + U_z h_{t-1} + b_z)$$

Now since we're more specifically interested in the effect that the near past and also near future affect the present's prediction of the future stock, we implement forward and backwards versions of the above, and join them together to get the following encoding of the corpus in the context of the near past and near future:

$$\overrightarrow{h}_t = \overrightarrow{GRU}(d_t), \qquad t = 1, \ldots, T$$
$$\overleftarrow{h}_t = \overleftarrow{GRU}(d_t), \qquad t = T, \ldots, 1$$
$$h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$$

Next-to-last comes the temporal attention layer, which operates in very much the same fashion as the news-level attention layer. This represents the entire sequence in a single vector as a weighted sum V:

$$o_t = \sigma(W_h h_t + b_h)$$
$$\beta_t = \frac{e^{\theta_t o_t}}{\sum_{t'} e^{\theta_{t'} o_{t'}}}$$
$$V = \sum_t \beta_t h_t,$$

where θ_t is the effect of o_i on the final stock value. Finally, we have a multi-layer perceptron (MLP) that takes V as input and outputs a softmax predictor $\hat{y}$.

5 Experiments/Results/Discussion

As a baseline, I implemented a multi-layer perceptron network that takes a sequence of news articles, averages their embeddings into a single 300-element corpus vector for each day, passes each subsequence of 10 corpus vectors through three hidden layers with 256, 128 and 64 units, respectively, and outputs a 3-unit softmax to signify the stock's direction: "UP" (RisePercent$(t) < -.41\%$), "DOWN" (RisePercent$(t) > 0.87\%$) or "STAY". Furthermore, I used dropout regularization with a rate of 0.5 for all three layers, and also employed Batch normalization. This ended up getting a training accuracy of 0.645 and a dev accuracy of 0.385 after 10 epochs under the Adam optimization algorithm, with a learning rate of 0.001 and a batch size of 200.

Unfortunately, I ran into some trouble when implementing the Recurrent layer. While training the model, I noticed that the training loss became "NaN" relatively quickly. I had a feeling that this was due to an "exploding gradient", so I attempted to use gradient clipping (range: [-1, 1]) to solve this issue. Unfortunately, even this range was not enough to keep the loss from becoming unreasonably high in terms of magnitude, and at the time of writing this sentence, it is infeasible to try anything else.

6 Conclusion/Future Work

The dense network performed reasonably well, especially since L2 regularization was not performed (hence the high variance), and the number of epochs was relatively low (hence the high bias). Regrettably, not much else came out of this project in terms of presentable results. If I had more time, I'd work on debugging the Dataset pipeline, which by far was the most challenging component of the project to work with. I learned a lot though, so that's got to count for something.

References

[1] Ryo Akita, Akira Yoshihara, Takashi Matsubara, and Kuniaki Uehara. Deep learning for stock prediction using numerical and textual information. In *Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference on*, pages 1–6. IEEE, 2016.

[2] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 261–269. ACM, 2018.

[3] Jianfeng Si, Arjun Mukherjee, Bing Liu, Qing Li, Huayi Li, and Xiaotie Deng. Exploiting topic based twitter sentiment for stock prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 24–29, 2013.

[4] Lipo Wang and Shekhar Gupta. Neural networks and wavelet de-noising for stock trading and prediction. In *Time Series Analysis, Modeling and Applications*, pages 229–247. Springer, 2013.

[5] Zhenkun Zhou, Jichang Zhao, and Ke Xu. Can online emotions predict the stock market in china? In *International Conference on Web Information Systems Engineering*, pages 328–342. Springer, 2016.