Predicting Stock Trends from News Articles

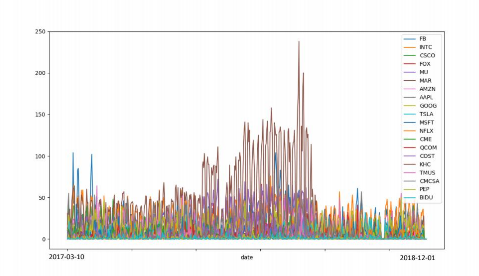
Sahil Nayyar, Electrical Engineering

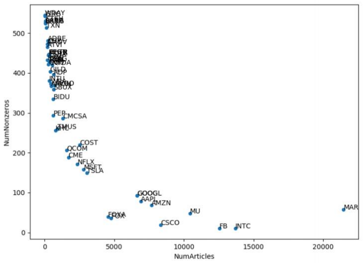
Predicting

The goal of this project is to predict stock trends from news articles. More specifically, the inputs are a time series of **news corpora**, chosen by their relevance to certain stock-holding entities, and the outputs are whether a stock's price will (1) **increase**, (2) **decrease** or (3) **stay the same**. In other words, this is a sequential trinary classification problem.

Data

About **2.5GB** worth of news articles were scraped from **Reuters.com** from 2017-03-10 to 2018-12-01. Those that mentioned the name of any of the top 20 NASDAQ-listed companies were kept; the others were discarded.





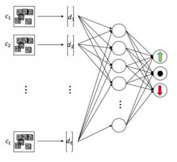
Features

The news articles were embedded using a pre-trained **Word2Vec** model [2]. Each article's words were embedded into vector form, then averaged. Words not present in the model's vocabulary were assigned a random embedding.

Models

These are the models that I'm currently working on (still debugging, unfortunately):

Baseline MLP

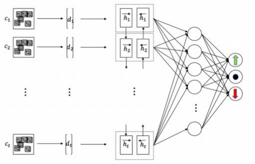


$$u_{ti} = sigmoid(W_n n_{ti} + b_n)$$

$$\alpha_{ti} = \frac{exp(u_{ti})}{\sum_j exp(u_{tj})}$$

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

Adding GRU Units

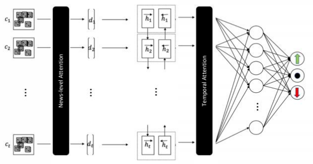


$$\overrightarrow{h_i} = \overrightarrow{GRU}(d_i), i \in [1, L]$$

$$\overleftarrow{h_i} = \overleftarrow{GRU}(d_i), i \in [L, 1]$$

$$h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$$

News/Temporal Attention Layers [1]



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

$$\alpha_{ti} = \frac{exp(u_{ti})}{\sum_j exp(u_{tj})}$$

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

$$o_i = sigmoid(W_h h_i + b_h)$$

$$\beta_i = \frac{exp(\theta_i o_i)}{\sum_j exp(\theta_j o_j)}$$

$$V = \sum_i \beta_i h_i$$

Results

To prepare the dad, I split it into a **train set of 488 dates** (2017-05-01 to 2018-08-31) and a **dev set of 61 dates** (2018-10-01 to 2018-11-30), chosen later since the goal is to predict *future* prices.

Unfortunately, I've run into many difficulties in implementing this project, so there's not much to show here right now.

Discussion

This has been a very challenging experience; in fact, the low error rate was what prompted me to choose it, since there's more to be learned from a greater challenge.

In the end, however, the results were not good: I spent way too much time collecting data, and although I had plenty of time to practice TensorFlow for my CS 221 project, I took for granted that there's a lot of non-transferable knowledge between what that project entailed (CNNs) and RNNs. Looking back, I should have started off quickly, using Keras to quickly prototype instead of waiting for the "right moment".

Future

My goal is to debug and train the three models described on the right in time for the final paper. If I had even more time, I would implement an algorithmic trading system that relies on the trained network – arguably this is a better test of its worth than just, say, accuracy.

References

[1] Hu, Z, et. al. "Listening to Chaotic Whispers: a Deep Learning Framework for News-oriented Stock Trend Prediction". *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* '18.

[2] Source: https://code.google.com/archive/p/word2vec/