Forecasting Stock Price Using Sentiment Analysis and LSTM Networks

Blake Hillier, Grace Li, Joe Puhalla

March 31, 2020

1 Overview

Forecasting stock prices is a widely known problem many people have attempted to solve through various models. A common approach is to use only numerical data to predict tomorrow's price. However, investors often use the news about the company's decisions, the public's attitude towards the company, and the state of the economy when developing trading strategies. In this paper, we propose a model using macro-economic variables to predict the future price of a stock, one of which is statements from the Federal Reserve about decisions on economic policies. Our model is comprised of XLNet to perform sentiment analysis on one macro-economic variable and an LSTM Neural Network to combine all the variables while capturing the effect time has on the future stock price. In this paper we first explain the ideas behind XLNet and LSTM Neural Networks. We then explain our implementation of XLNet and LSTM and calculate their mean squared error (MSE) using appropriate subsets of our data. Finally, we explain how we use them together to forecast the stock price and calculate the entire models MSE before making some concluding remarks.

2 XLNet

XLNet is an autoregressive pretraining approach for NLP models. Pretraining a model is used to teach a model a broad field before being applied to a specific problem. This allows it to pick up potential nuances within the data and then apply this knowledge to better understand the specific problem. It also allows copies of the same model to be fine tuned to different problems, decreasing the total learning time. Autoregressive pretraining approaches create a conditional probability distribution based on the likelyhood function

$$p(x) = \prod_{t=1}^{T} p(x_t | x_{<t})$$

which only sees the relationship between previous text (it can also be modified to only see the relationship between text after the word). This is problematic since most words derive their meaning from context at both the beginning and the end of a sentence. XLNet solves this problem by calculating a distribution based on all the other text in the word by maximizing

$$\max_{\theta} E_{z \sim Z_T} \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} | x_{z<t}) \right] = E_{z \sim Z_T} \left[\sum_{t=1}^{T} \log \frac{e^{g_\theta(x_{z<t}, z_t) l(x_t)}}{\sum_{x'} e^{g_\theta(x_{z<t}, z_t) l(x')}} \right]$$

where Z_T is the set of all permutations of text of length T, $z \in Z_T, x_{z<t}$ is the sequence of text from 1 to $t-1$, and g_θ transforms x to a sequence of hidden words with the first $t-1$ set of words as additional information. They note the permutations don't affect the actual order of the text sequence, just the factorization of the likelihood function. Because this is based on the likelihood function, it removes the idea each hidden text is independent of the others while still maintaining the benefits through encoding due to g_θ. In order for g_θ to accomplish this, they split it into two different transforms: the query g_θ which looks at the first $t-1$ words in the permuted order to predict the t^{th} word, and the content h_θ which simply encodes the first t words in the permuted order. The one downside to this model is the complexity of the optimization leading to a slower convergence. To reduce this problem, they adjust the equation to

$$\max_{\theta} E_{z \sim Z_T} \left[\log_{p_\theta}(x_{z>c} | x_{z \leq t}) \right] = E_{z \sim Z_T} \left[\sum_{t=c+1}^{|z|} \log p_\theta(x_{z_t} | x_{z<t}) \right]$$

which changes the model to only predict the last $T - c$ words in the permutation order, where $c \approx \frac{T(1+K)}{K}$ for some parameter K.

3 LSTM

In this paper, the LSTM neural network is uesed to predict the stock price, the input data is the historical stock price and sentiment analysis results. Here, the sentiment based LSTM neural network (named sentiment-LSTM) is aimed to minimize the following loss function:

$$\ell = min \sum_{t=p+1}^{p+T} \left\| X_t - \hat{X}_t \right\|_2^2$$

Where T denotes the number of prediction time slots, i.e, $t = 1, ..., p$ are the observations (training input data), $t = p+1, ..., p+T$ are the predicts (training output data); and $\hat{X}_t$ is given as following:

$$\hat{X}_t = \alpha X_t^A + \lambda S_t^A + c = \alpha X_t^A + \lambda f_2 \underbrace{((S_{t-i})_{i=1}^{p})}_{\text{Sentiment}} + c$$

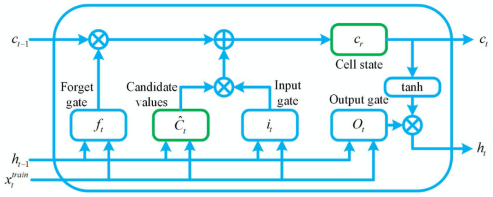

Figure 1: LSTM Procedure

Denote Formula as the training input data. Figure 1 shows the LSTM's structure network, which comprises one or more hidden layers, an output layer and an input layer. LSTM networks' main advantage is that the hidden layer comprises memory cells. Each memory cell recurrently has a core self-connected linear unit called " Constant Error Carousel, which provides short-term memory storage and has three gates:

Input gate, which controls the information from a new input to the memory cell, is given by

$$i_t = \sigma(W_i \times [h_{t-1}, \chi_t^{train}] + b_i)$$
$$\hat{c}_t = tanh(W_c \times [h_{t-1}, \chi_t^{train}] + b_c)$$

h_{t-1} is the hidden state at the time step $t-1$; i_t is the output of the input gate layer at the time step t; $\hat{c}_t$ is the candidate value to be added to the output at the time step t; b_i and b_c are biases of the input gate layer and the candidate value computation, respectively; W_i and W_c are weights of the input gate and the candidate value computation respectively; and $\sigma(x) = 1/(1+e^{-x})$ is the pointwise nonlinear activation function.

Forget gate, which controls the limit up to which a value is saved in the memory, is given by

$$f(t) = \sigma(W_f \times [h_{t-1}, \chi_t^{train}] + b_f)$$

where f_t is the forget state at the time step t, W_f is the weight of the forget gate; and b_f is the bias of the forget gate.

Output gate, which controls the information output from the memory cell, is given by

$$c_t = f_t \times c_{t-1} + i_t \times \hat{c}_t$$
$$o_t = \sigma(W_o \times [h_{t-1}, \chi_t^{train}] + b_o)$$
$$h_t = o_t \times tanh(c_t)$$

where new cell state c_t are calculated based on the results of the previous two steps; o_t is the output at the time step t; W_o is the weight of the output gate; and b_o is the bias of the out put gate.

4 Experiment

Because our model has multiple core algorithms, we need to calibrate each main component individually as well as when they are all together to ensure we are obtaining an optimal accuracy. This section will discuss the data we use, how each algorithm performs on their own, and how they perform when put together.

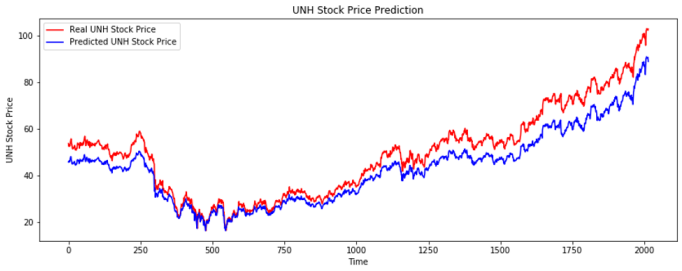

Figure 2: Predicted vs. Actual Stock Price

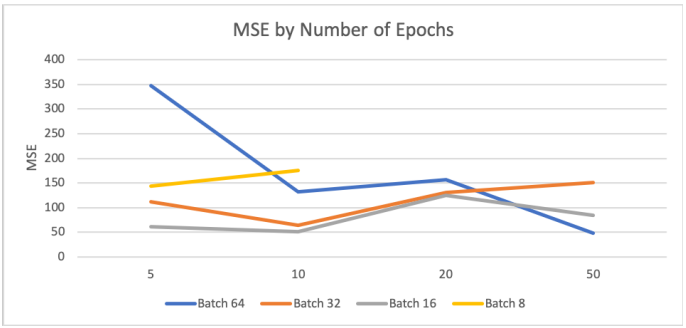

Figure 3: MSE by Number of Epochs

4.1 Data

Our data consisted of macroeconomic data and the previous stock priced. The macroeconomic data consisted of the quarterly GDP, monthly CPI, monthly unemployment rate, monthly inflation rate, the ten-year treasury rate, the 1 year LIBOR daily, and the statements from the Federal Reserve. All the data ranges from the beginning of the year 2000 to the end of 2014. Since our stock is daily data, all previous data were duplicated into daily data and aligned with the stock information.

4.2 Sentiment Analysis

We used pytorch's implementation of XLNet-base for our model. Sentiment was assigned to the Fed's Statements by looking at the percent change of the UNH stock. If the stock increased, the sentiment was positive, and negative otherwise. We then tested the accuracy of the XLNet by randomly selecting 80% of the data to train, and the remaining 20% to test. This was done using a GPU on Google's Colab. We had to carefully select our batch size, maximum statement length, and epoch combinations carefully due to memory constraints. This posed an interesting problem for us since a larger maximum statement size allows more of the Fed's statement to be considered for sentiment, but requires a smaller batch size resulting in severe over fitting. A smaller statement size reduces the amount of information available, but allows a bigger batch size leading to better generalizations. After some testing we found a maximum statement length of 128, batch size of 24, and 10 epochs produced the best accuracy of 77.1%.

4.3 LSTM Stock Prediction

The LSTM network we use includes four layers each with a dropout rate of twenty percent. The layers are of size $256, 128, 64, 32$ and then the final layer. We sampled through several activation functions, starting with linear activation, and determined the ReLu activation to provide the best accuracy of the model. In order to determine the accuracy of the model, we plot both the predicted and actual stock price values of UNH. We then determine the mean squared error between the predicted and actual prices. We sampled across several amounts of epochs $(5, 10, 20, 50)$ and batch sizes $(8, 16, 32, 64)$ in order to determine the best architecture for the model. For simplicity purposes, we used the already given 'adam' optimizer. Our feature set for the model were daily measures of price, GDP, CPI, inflation, unemployment, LIBOR, and the 10-year treasury yield. GDP data was given quarterly and thus had to be converted to daily data. In addition, CPI, inflation, and unemployment were given as monthly data and were converted to daily data. Our train and test data were split 50-50. With a batch size of 8, the 20 and 50 epoch sized networks caused memory issues and thus were unable to be completed. A batch size of 64, and 50 epochs yielded the lowest MSE among all models (2). More tests need to be conducted in order to determine the most accurate model. Furthermore, the number of layers needs to be varied as well as the optimizer used. Tests need to be done to determine which features contribute the most to the predicting power of our model. Sentiment scores of the FOMC transcripts also need to be added as features of the model. The preliminary results provide for a proof of concept for our model and give promise to further tests with more features.

4.4 XLNet and LSTM

Using the optimal model parameters through the previous sections, we created a pipeline using both models. We first trained the XLNet on the entire text data, and then predicted the sentiment on the same dataset. This was then merged with the input data for the LSTM, and was trained using a portion of the stock data. Once trained, we validated it with the last 2014 data points to obtain the MSE: 25.226. This is lower than our previous tests with the LSTM, showing the capability of XLNet improving our forecasting accuracy. Figure 4 shows our test case after training with XLNet and the LSTM model.

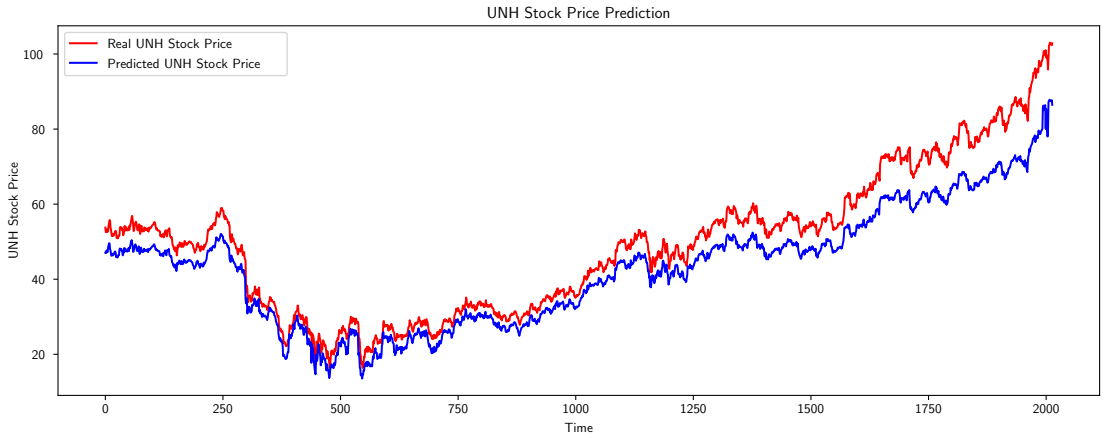

Figure 4: Forecast with XLNet and LSTM compared with the actual price

5 Conclusion

In this paper we attempt to build a stock forecasting model using sentiment analysis with XLNet and then forecast using an LSTM network. Even though our data only contained a few features, of which all but one were macroeconomic, we still obtained results showing promise with this model structure. One big downside of our model is the variance in the accuracy of the XLNet. While it is possible to get 77.8%, when training the model we would also often get accuracy's near 60%, or even 40%. We believe this is due to our low amount of data, and hope it will also improve with a better sentiment measure. Using a GPU with more memory would allow us to increase our maximum statement length which could also improve XLNet's accuracy. We also hope by carefully selecting more features, including micro-economic features, we can increase our models accuracy, and plan on increasing the time duration of our data set. Lastly, while judging the MSE of the predicted value is a great way of understanding the model's accuracy, simulating a trading strategy using the model would allow us to more concretely understand how usable our model really is. For a full look at the code, check out our github.

References

[1] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregresive Pretraining for Language Understanding*. Carnegie Mellon University, 2Google AI Brain Team, Jan 2^{nd} 2020.

[2] Xinyi Li, Yinchuan Li, Hongyang Yang, Liuqing Yang, Xiao-Yang Liu. *DP-LSTM: Differential Privacy-inspired LSTM for Stock Prediction Using Financial News*. Columbia University, Beijing Institute of Technology.

[3] Github LSTM Stock Prediction, https://github.com/laxmimerit/Google-Stock-Price-Prediction-Using-RNN---LSTM/blob/master/Google%20Stock%20Price%20Prediction%20Using%20RNN-%20LSTM.ipynb