Stock Trend Prediction using Deep Learning

Deep Learning Project Report

Github Link : https://github.com/bigholdlu/Stock_Prediction_Sentiment_Analysis

Team members : Anshu Mathur (am10263), Jeremy Lu (xl2810)

1. Introduction

Algorithmic trading has always been a hot topic in both the financial area and deep learning area. In the deep learning area, the majority of focus is on price predictions. However, performance of the stock market has a deep relationship with the confidence of investors and therefore, lots of uncertainty is buried in the price chart. Financial news is an important reference for investors to analyze if a stock has potential profit. We came up with the idea of automating the process of analyzing the financial news to forecast the trend of the stock market. We want to research on the relationship between news and market trends and potentially make predictions with deep learning models.

Objectives

- To find the relationships between news headlines and market trend, here news headlines are financial news from major press and market trend is numerized by *Dow Jones Industrial Average(DJIA)*.

- To implement deep learning models, such as LSTM and BERT, that intake financial news and predict market trend with reasonable performance. Model performance will be evaluated with prediction accuracy.

2. Literature Survey

We performed a broad research in implementation of deep learning in the stock market. There are mainly three types of researchs under this topic : price prediction, trading strategy and sentiment analysis.

Price Prediction is the most common type of research and is very straightforward. The goal is to directly predict the stock price with a historical price chart. Price charts are used as time series for input. Popular choices of model are RNN based time series prediction models because of its' 1d processing nature. However, some research projects have used CNN as a model which involves converting 1d time series into 2d image-like input. Drawback of this method is $overfitting$. Price data tends to be very uncertain due to unpredictable market events, and due to several other contraints, it is not a very efficient method.

Trading strategy involves training a trading agent using reinforcement learning. The goal is to maximize the return buy search for the best time for buying or selling. The trading agent gets a reward for buy low sell high and a penalty for the opposite. The policy becomes very complicated when fraction trading is allowed. Even though the published result is profitable, if the trading agent is highly profitable, it would violate the fundamentals of the market and thus, become invalid quickly.

Sentiment analysis performs sentiment analysis of financial news and forecasts the trend accordingly. A research paper by Kalyani and Bharathi built classfication model on predicting trend with headlines. They selected statistical learning model including Random Forest, Naive Bayes and SVM classifier. Raw headlines were tokenized into word vectors and assigned with binary labels. All the three models had good accuracies of about 80 percent.

We plan on doing $Sentiment\ analysis$ in our project since it is a more accurate and a reliable method to predict the trend of stock market, by using the LSTM and BERT models to perform this prediction and to analyze the results.

2.1 LSTM Model (Long Short-Term Memory Network)

The LSTM network is a recurrent neural network and is trained using Backpropagation. It has memory blocks which are basically connected through layers. It also prevents the problem of disappearing gradient. It is a nice model to analyze time-series data and hence, will be useful in achieving our objective of this project. LSTM is a very important RNN model since it can memorize all the long and short-term values and hence, is very allows the neural network to retain the necessary information only.

The news headlines play an important role in predicting the trend of the stock market. So, in our LSTM model, we will have the news headlines as our input. Our output will be the predicted DJIA values. We will use the $Sequential$ way of building the deep learning model.

Our input X (train_X) is the news headlines, input Y (train_Y) is the labels generated from the djia values (shown in code), the labels have been tagged into 3 classes - 0, 1 and 2 based on the djia values. Our aim using LSTM is to predict in one which of the classes would a stock be classified into based on news headline as the input.

2.1.1 Brief Architecture

The first layer in our LSTM is the $Embedding\ Layer$**.** It will convert our tokenized words into useful embedding vectors. It takes three parameters as an input : the total tokenized words, hidden size and the length of the input sequences.

For the second layer in our LSTM, we take $units$ = 100, which is the dimensionality of the output space.
$dropout$ is taken to be 0.2, since it tells us what fraction of units should be dropped for linear transformation of the inputs. We add the dropout layers to prevent overfitting in our model.

The next layer in our LSTM is a $softmax\ layer$ with output units = 3 ,we have taken this value to be 3 because we want to classify our stock into one of the given 3 classes to predict the trend of that stock.
We take the activation to be softmax since it is used to predict a multinomial probability distribution and hence, we will have one output value corresponding to each node in the output layer.

After building the layers of the LSTM, we compile our model by setting the $loss\ =\ categorical\_crossentropy$.
We want to find accuracy, so we set the $metrics\ =\ accuracy$ and the $optimizer\ =\ Adam$, we use Adam optimization since it is a stochastic gradient descent method. We finally tokenize the validation data also, and train the model using $model.fit$ method, where we take the number of epochs = 20, batch size = 32 and validation data as the tokenized validation data.

We finally perform $model.predict$ on the tokenized test data to predict the trend of DJIA. Since we want a list of the predicted model trend (classified into 3 classes), we use $argmax$ to print the category / class in the list of the predicted model trend.

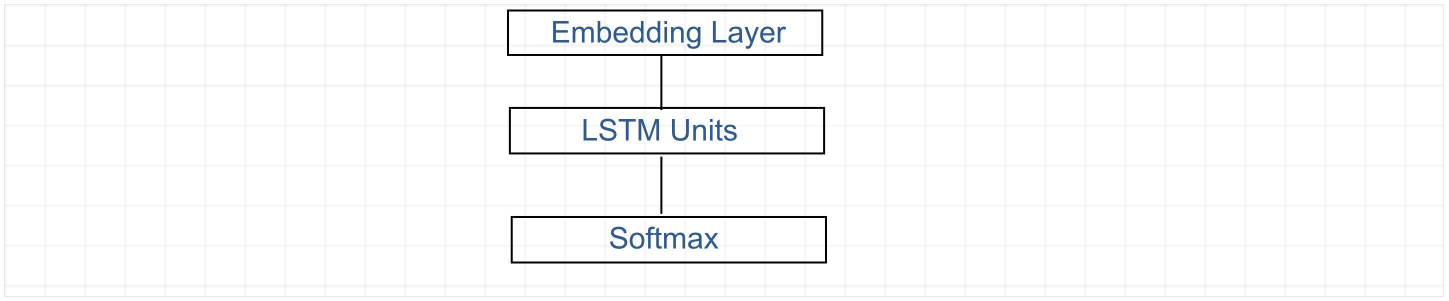

Figure 1 : LSTM Network

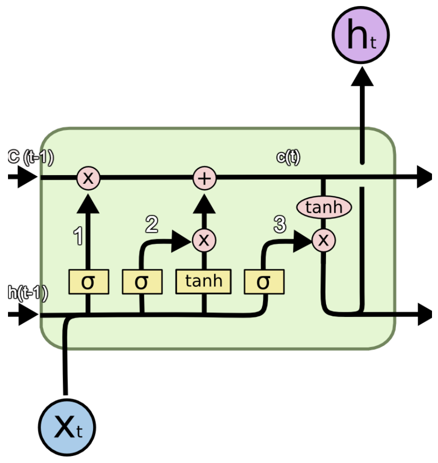

Figure 2 : LSTM Unit

2.2 BERT (Bidirectional Encoder Representations from Transformers) Model

BERT is a transformer model that uses encoder to encode the relation between words. In order to synchronize the length of the input, an attention mask is generated for each input that represents the index of valid input. Encoder reads the sequence bidirectionally when preprocessing input. The output from model is decoded into actual prediction.

Below chart is a single transformer model and multiple transformers can be stacked to create for more complex tasks. An important feature of BERT model is to tokenize the string input with attention mask. Specifically in this project, the nature of string input makes data to have different length. The tokenize procedure add padding to the strings to make them into a same length. Noted the length is a hyperparameter defined in later section 3.4 and should be long enough to cover all information from input. An attention mask is generated for each tokenized string. The attention mask acts as a layer before the MLP learning, thus there is no trainable parameters in it.

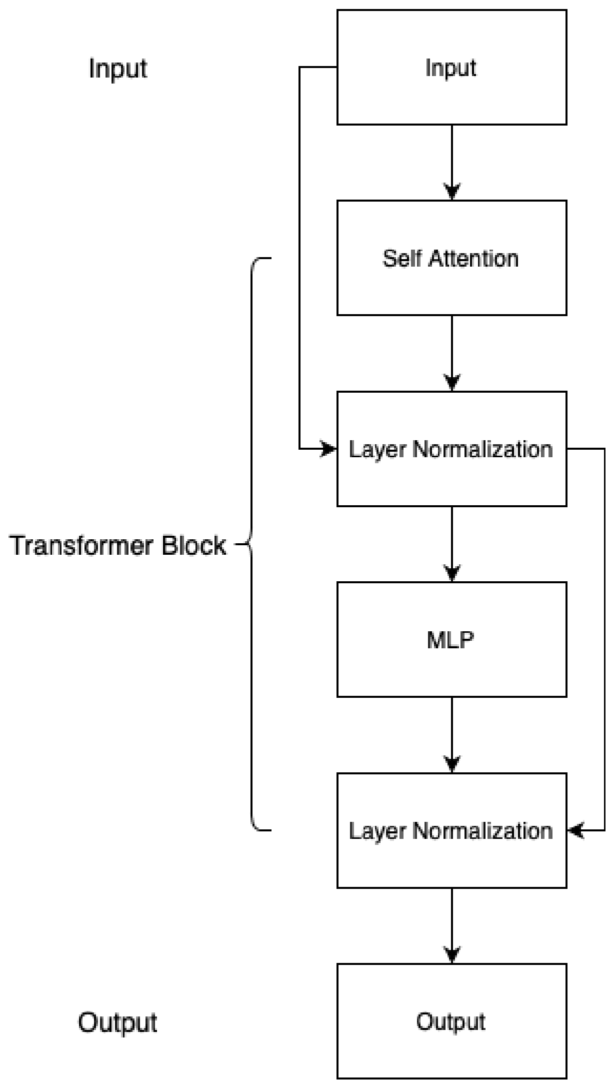

Figure 3 : Transformer Architecture

2.2.1 Pipeline of the project using BERT

Below is the pipeline of BERT model. There are three major steps with each step separated into smaller procedures

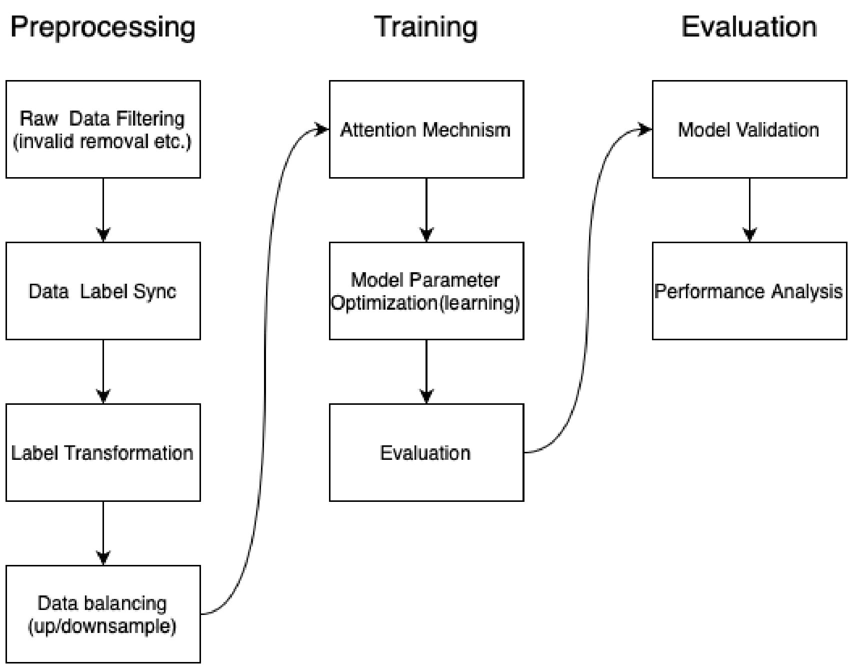

Figure 4 : BERT Model Pipeline

3. Technical Details

3.1 Input and Output

The *input data* for the pipeline contains two dataset, namely two news headline dataset with time stamp for each headline, and a timestamped price history. The *output* for the model is a single number representing the trend.
- For BERT, the input is a string of single headlines and output is a 3x1 tensor representing the probability for each class.
- For LSTM, we convert the training labels to one-hot encoded vectors.

3.2 Data Preparation

Raw data filtering: This is the first step in our data preprocessing. Both headline datasets and price history are filtered by dropping invalid value. For price history, the ratio of daily close and open are calculated. The ratio are converted to class labels accoridngt to threthold limit which choice is mentioned in Hyperparameter selection. Dataset and price are then combined with common time using inner join method.
Data imbalance: We then inspect the class distribution by ploting histogram of all classes. We noticed class '1' has significantly higher appearence than other classes. Therefore we performed upsamping on rare class and down sampling on major class to balance the dataset. Noted in order to test the effect of data imbalancing, we performed a train test with imbalanced data and we noticed overfitting and extremely low prediction accuracy.

3.3 Training details and Hyperparameter Selection

General :

Positve Trend Threshold: 1.007

Negative Trend Threshold: 0.993
Upsampling factor: 2
Downsampling factor: 2
Input dimenstion: 160x1
Output dimension: 1x1

LSTM

- optimizer = adam
- no. of epochs = 20
- batch_size = 32

First Layer :

- hidden_size = 10
- length of input sequences = token_list.shape[1] = 30

Second Layer :

- units = 100
- dropout = 0.2
- recurrent dropout = 0.2

Third layer :

- no. of output units = 30
- activation = softmax

BERT

- Encoder maximum length(input vector size): 160
- Batch Size: 16
- Epoch: 15
- Optimizer: AdamW
- Learning Rate: 2e-5
- Dropout: 0.3

3.4 Loss Functions

3.4.1 For BERT

We choose *CrossEntropy loss* as loss function because we perform classification with output of probability. Cross-entropy loss increases when predicted probability diverges from true value. Since we have three class for prediction, multi-class cross-entropy loss is used, which is calculated as:

$$crossentropy\ loss = -\sum_{i=1}^{output\ size} y_i \cdot \log p_i$$

$$where\ p_i\ is\ the\ probability\ of\ prediction,$$

$$y_i\ \ is\ the\ corresponding\ target\ value,$$

$$and\ output\ size\ is\ the\ number\ of\ scalar\ values\ in\ the\ model\ output.$$

3.4.2 For LSTM

We have chosen our Loss function as *Categorical Crossentropy Loss* as we use it in cases where there are multiple classes out of which only one is true. Since in our model, we have 3 output classes, out of which one can be true, i.e. a stock can either be classified into classes 0, or 1, or 2 correctly.

$$categorical\_crossentropy\ loss = -\sum_{i=1}^{output\ size} y_i \cdot \log \widehat{y}_i$$

$$\textit{where } \widehat{y}_i \textit{ is the } i^{th} \textit{ scalar value in the model output,}$$
$$y_i \textit{ is the corresponding target value,}$$
$$\textit{and output size is the number of scalar values in the model output.}$$

4. RESULTS

4.1 BERT model

The first two plots show the training vs validation accuracies and losses across all epochs. The descending training loss and ascending accuracy shows that the model is learning from training dataset. It is interesting that even though accurracy of validation increases as model learns, loss of validation also increases. This might be due to some extent of overfitting in the model but the accuracy increases on the validation as well as the test data sets.

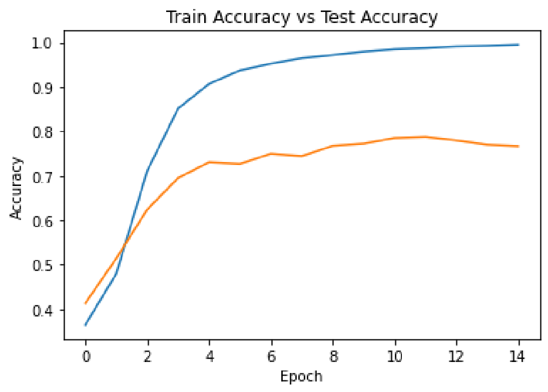

Figure 5 : Train and test accuracy

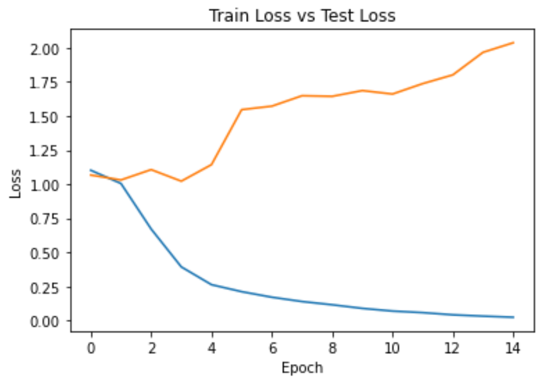

Figure 6 : Train and test loss

Figure 7 is a heatmap of prediction on validation dataset. We can see that classification of labels 0 and 2 are better. The label 1 has lower contrast and given similar number for each class, class 1 has significantly lower number of correct predictions. Also, in the case of the wrong predictions of class 0 or class 2, the classifier tends to classify to class 1 than to opposite class. Hence, class 1 is harder to predict as compared to the other 2 classes.

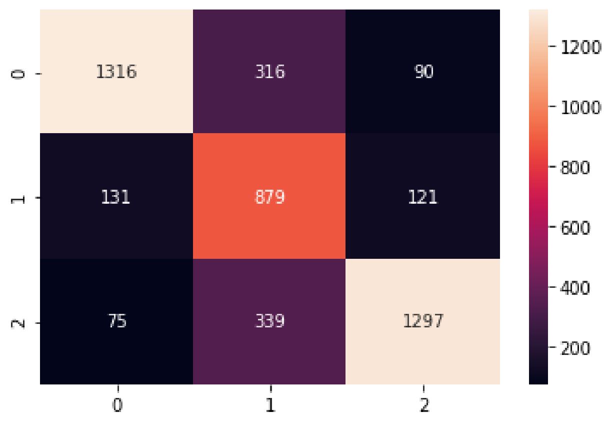

Figure 7 : Heatmap of prediction on validation dataset

From the classification report given below, we can see that precision are stable across different class. While recall rate for positive and negative class are over 85 percent, recall of stable class is relatively lower. The reason can be the less clear relationship between news to stable market. The overall accuracy of the prediction shows that the model is able to predict trend of stock market with an acceptable accuracy of 77 percent.

Label	Precision	Recall	F1-Score	Support
Negative, '0'	0.76	0.86	0.81	1522
Stable, '1'	0.78	0.57	0.66	1534
Positive, '2'	0.76	0.86	0.81	1508
Accuracy	0.77			

Table 1: Classification report for validation dataset

4.2 LSTM model

The first two plots show the training vs validation accuracies and losses across all epochs. We can see that the accuracy increases as the training is performed on the model. We can observe that even though accurracy of validation increases as the model learns, loss of validation also increases. This might be due to some extent of overfitting in the model but the accuracy increases on the validation as well as the test data sets.

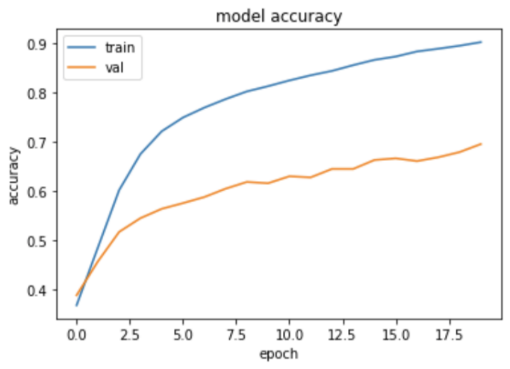

Figure 8 : Train and test accuracy

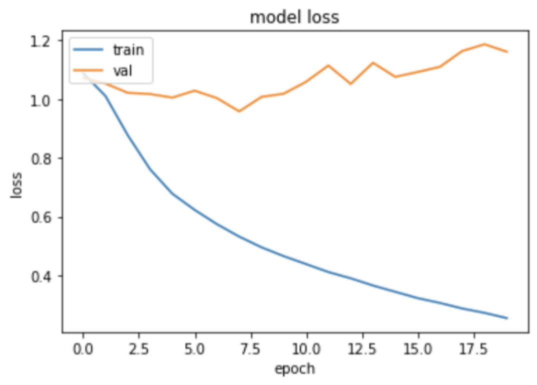

Figure 9 : Train and test loss

Figure 10 is a heatmap of prediction on validation dataset. Similar to BERT results, we can see that the classification of label 0 and 2 are better. Class 1 has significantly lower number of correct predictions.

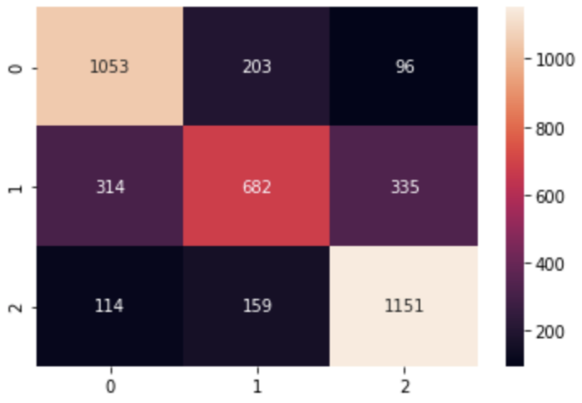

Figure 10 : Heatmap of prediction on validation dataset

Table 2 shows the classification report for LSTM. The overall accuracy of the prediction shows that there is model is able to predict trend of stock market with an acceptable accuracy of 70 percent, which is slightly lower than accuracy of the BERT model.

Label	Precision	Recall	F1-Score	Support
Negative, '0'	0.78	0.71	0.74	1481
Stable, '1'	0.51	0.65	0.57	1044
Positive, '2'	0.81	0.73	0.77	1582
Accuracy	0.70			

Table 2 : Classification report for validation dataset

4.3 Comparison between the two models

Both models are able to relate news and DJIA trend. Comparing heatmaps from the two models above, we can see BERT is better at recalling negative and positive class, but less effective at recalling the class '1', LSTM gives a stable

recall rate. From perspective of precision, BERT performs better with stable and high performance for all classes. From F-1 Score, we can see both models find the label '1', the label of no market trend, is harder to classify than label of clear trend.
Since in stock market, identifying all classes are important, we choose accuracy as the best measurement of performance. We can see from the results above that BERT model gives a better accuracy as compared to the LSTM model.

5. DISCUSSION AND IMPROVEMENT

Both models find clear relation between news and price trend. But we also noticed that both the models are better at classifying clear trend of rise or fall than no rise or fall (stable trend). A possible reason can be characteristic of the stable class. For stable class, the news might not have relationship with market trend, thus can be considered as $out\ of\ distribution\ data$.
We propose two ways to solve this problem as an improvement plan:
$First$ is to use confidence score. It is similar to the labeling process of price trend using threshold rate of return. Instead of three classes, we propose to use binary labels and add threshold limit at tensor output. Only probabability higher than certain values will be classified into classes.
Another method is to use small prubation to training data. However, a small prubation can be added to each input to help in $out\ of\ distribution\ data$ detection

6. CONCLUSION

We goal was to find relation between news and a market trend with sentiment analysis, and ffinally to predict the trend with headlines. We implemented two models- BERT and LSTM, trained them and tested with a 3 year long dataset. We achieved accuracies of 77 percent and 70 percent for the BERT and LSTM models respectively. We can conclude from the results that both the models are better at predicting changing (positive / negative) trend as compared to predicting the stable trend. As we can see from the results that since BERT has a higher accuracy than LSTM, it is a better model in predicting the market trend using news headlines.

7. REFERENCES

[1] Tul, Qurat, et al. “Sentiment Analysis Using Deep Learning Techniques: A Review.” International Journal of Advanced
Computer Science and Applications, vol. 8, no. 6, 2017. Crossref, doi:10.14569/ijacsa.2017.080657.
[2] K. A. Althelaya, E. M. El-Alfy and S. Mohammed, "Evaluation of bidirectional LSTM for short-and long-term stock market prediction," 2018 9th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 2018, pp. 151-156, doi: 10.1109/IACS.2018.8355458.
[3] Ugur Gudelek, S, Arda Boluk, Murat Ozbayolu, Department of Computer Engineering Ankara, Turkey, A deep learning based stock trading model with 2-D CNN trend detection.
[4] Kalyani J, Departmen of Computer Engineering, KJSCE, Mumbai, Bharathi H. N, Departmen of Computer Engineering, KJSCE, Mumbai, Prof Jyothi Rao, Departmen of Computer Engineering, KJSCE, Mumbai, Stock Trend Prediction Using News Sentiment Analysis
[5] Jiefeng Chen, Yixuan Lu, Xi Wu, Yingyu Liang, Somesh Jha, Robust Out-of -dsitribution Detection for Neural Networks.
[6] https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/
[7] https://keras.io/api/layers/core_layers/embedding/