Financial Time Series Analysis of Stock Data

Shengnan YU

May 2018

Contents

1 Introduction

In the recent years, big data become a popular topic in many fields around the world. People keep studying on the applications of big data to improve service and life. In many aspects with high dimensional data and complicated behaviours, the traditional machine learning and statistical approaches have limitation to construct a model to predict the target precisely. Deep learning is a relatively new technology, which uses neural networks to simulate the work of humans' brains to learn things from data.

Time-series prediction is a common techniques widely used in many real world applications such as weather forecasting and financial market prediction. It uses the continuous data in a period of time to predict the result in the next time unit. Many time-series prediction algorithms have shown their effectiveness in practice. The most common algorithms now are based on Recurrent Neural Networks(RNN), as well as its special type - Long-short Term Memory(LSTM) and Gated Recurrent Unit(GRU).

Stock market is a typical area that presents time-series data and many researchers study on it and proposed various models. In this project, a simple multi-layered LSTM model and a dual-stage attention based LSTM model are used to predict the stock price. The following chapters will introduce the detailed models, implementation and test result.

2 A Multi-layered LSTM Model

2.1 Background

As can be known that, traditional artifical neural network assume that all the inputs are independent to each other, which is unexpected for time-series prediction. However, RNN apply same operations on every element in the series and every operation is dependent on result calculated previously. In other words, RNN can memorize former information in the several past steps.

RNN has a limitation that it can not learn and connect the information between very large gap in the series. Therefore, a specil improved network, LSTM is introduced to

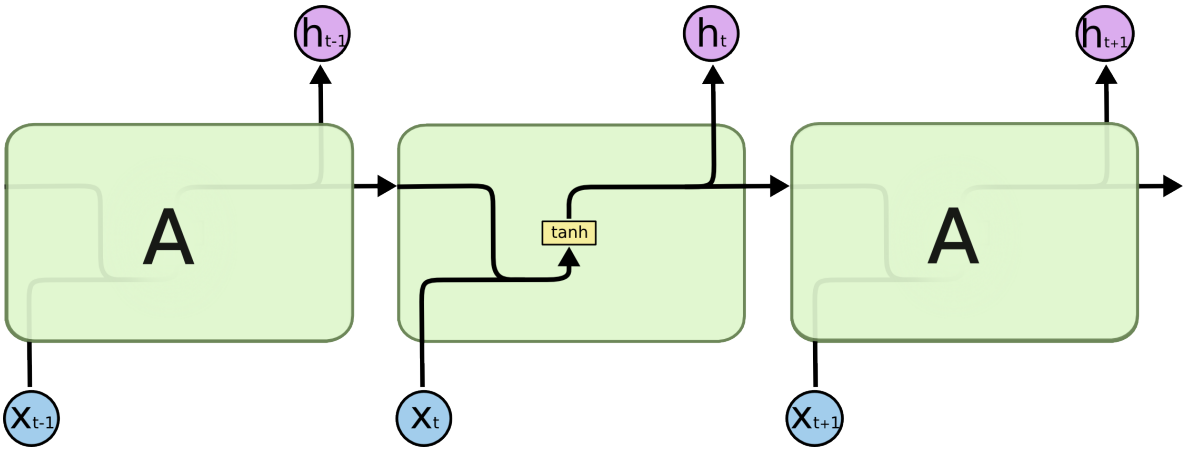

Figure 1: Repeat module in RNN

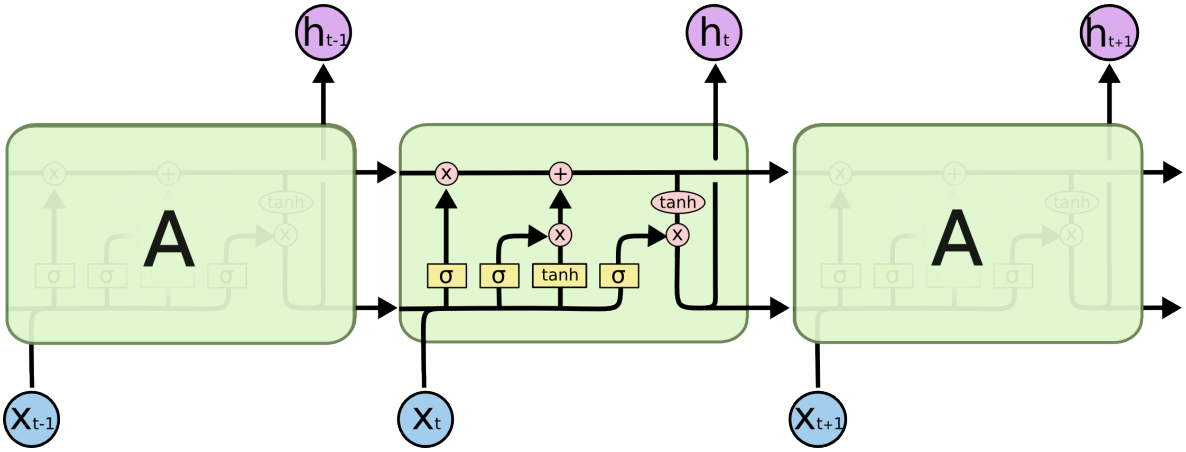

Figure 2: Repeat module in LSTM

solve this problem. LSTMs have the same chain like structure like RNN but different repeat module structure. Instead of RNN's single neural network layer shown in Figure 1, there are multiple layers interacting in a special form, which is presented in Figure 2.

The most significant things in LSTMs is the cell states, which stands by the horizontal line at the top part of Figure 2. The cell state behaves like a conveyor belt so it keeps running in the chain of LSTMs with some linear functions transformation, which retains the previous information. The LSTM allows three gates to add or remove information to the cell state and those gates are commonly formed by an activation function layer(usually sigmoid) and a point-wise multiplication operation.

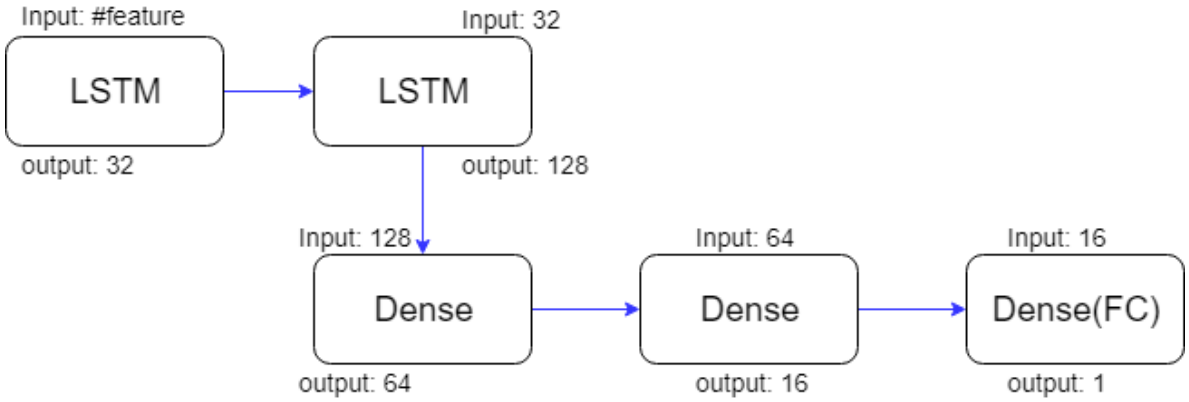

Figure 3: 5-layer LSTM model

2.2 Model

The first try of model is a simple one with stacked layers. The model consists of 5 layers in total where the first two are LSTM layers and the remaining are dense layers. The structure is shown in Figure 3. As can be seen, the first LSTM layer use the features of data as input, in this project task, OHLCV(open, high, low, close, market value) can be regarded as the input features so the input has $n \times T \times 5$ dimensions(for simplicity, only 5 is used because $n \times T$ is general in all vectors), or 1 if only close price is used. Then the second LSTM layer accept a dimension-32 vector and output a 128 one. Then there will be three linear layers, which gradually dense the vectors into dimension 1.

3 A Dual-stage Attentioned based LSTM Model(DA-LSTM)[3]

3.1 Background

3.1.1 Encoder-decoder model

Encoder-decoder model is proposed to solve seq2seq problem, which accepts an sequence as input and output another sequence[1]. It has become popular in many applications especially for natural language processing. The model processes data in two stages. As

shown in Figure 4, encoder is used to transfer the input sequence to a fixed-size vector while decoder is to convert the vector back to a sequence.

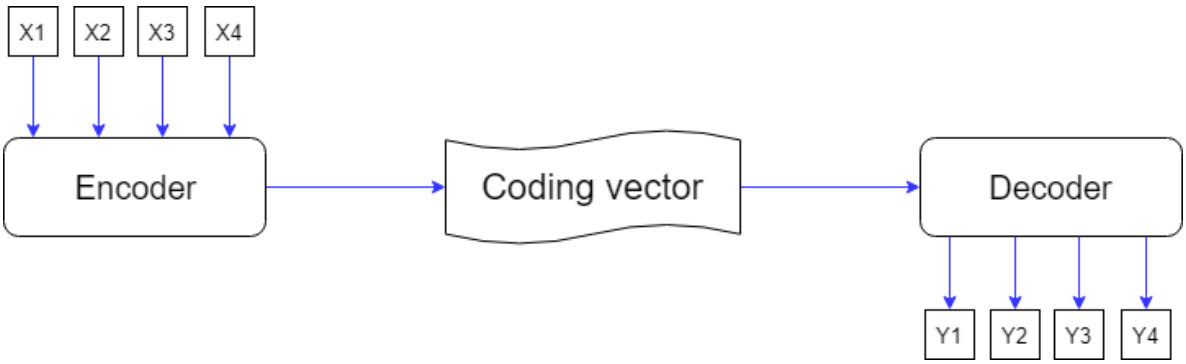

Figure 4: Encoder-decoder model

Encoder and decoder can be implemented with various models such as CNN/RNN/LSTM/GRU and any customized combinations. Although te encoder-decoder model is classical and widely used, it has weakness to limit its effeteness. The most significant one is that the only contact between two phases is the coding vector, which means encoder needs to compress all information from input to a fixed-size vector. Such processing methods has two shortcomings, one of which is that the coding vector may not represent all the semantic information from input sequence. The other one is that the information coming ahead will be lighten or overridden by that coming later. The sequence length longer, the problem severer. Therefore it actually affects the precision of the model.

3.1.2 Attention

To solve the problems of encoder-decoder model, an attentioned mechanism][2] is proposed by researchers. When the model is going to output, it will generate a attention range to emphasize the part which should get "more attentioned" in the sequence. The output is produced based on attentions and repeat previous procedure. The schematic diagram is shown in Figure 5.

The new attention based model does not require encoder process all the information in to a fixed-size vector. It encodes the input sequence to a vector sequence. In the

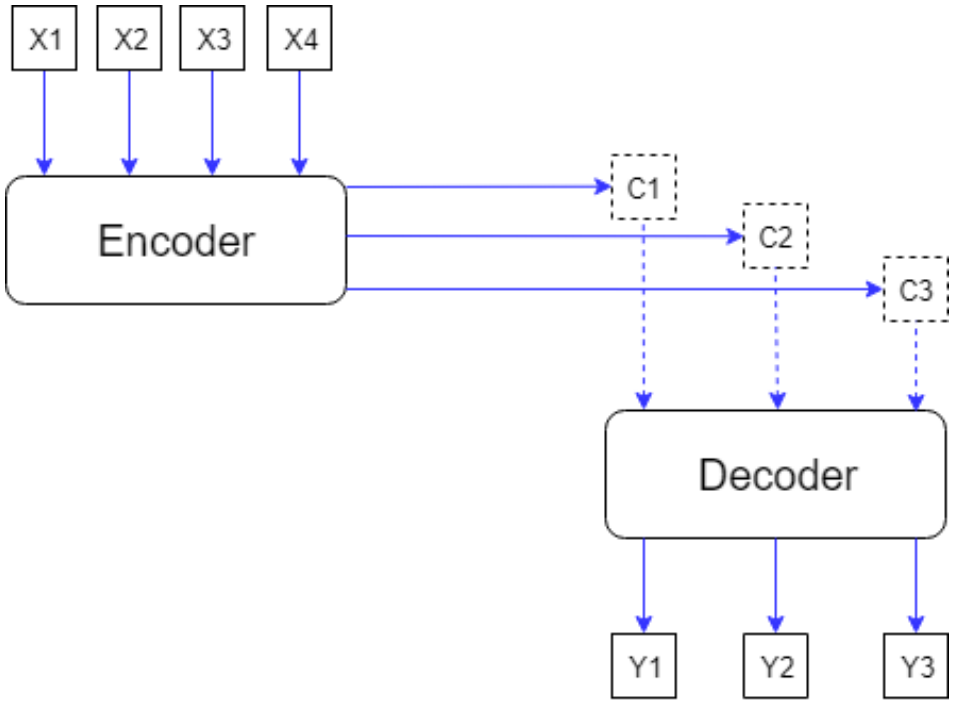

Figure 5: Attention mechanism

second phase, the decoder will pick a sub-sequence from the vector sequence and process. Therefore every output makes full use of information in the input sequence, which results into a more accurate model.

3.2 Detailed Model

The core of the project is to combine the encoder-decoder model with attention mechanism to handle the finical time series prediction. As mentioned before, the model will be divided into two stages - encoder and decoder. Meanwhile, in the encoder, a novel idea is that the input uses a driving time series. For example, if we are going to predict the stock price of AAPL.US, the stock price of Microsoft, Alphabet and other tech companies can be the driving series because they may have influence on Apple Inc. stock. In the decoder phase, the Apple stock price is input with the encoder hidden state - coding vector. A temporal attention automatically select the relevant components in the coding vector. With these definition, the model can extract the useful information as much as possible. The model structure is illustrated in Figure 6.

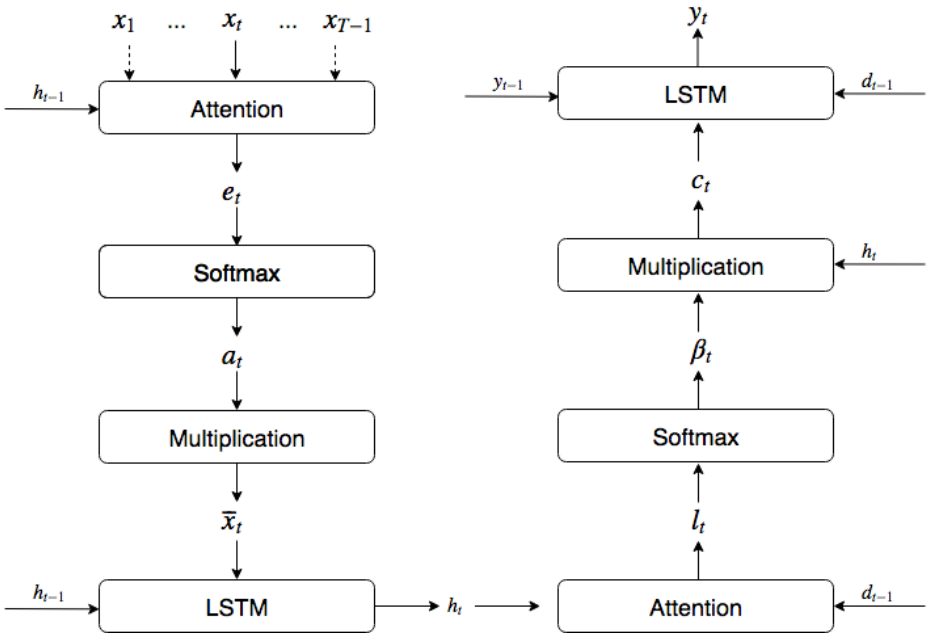

Figure 6: Model structure

3.2.1 Encoder

In the encoder, the dimension of input is $n \times T \times m$, where m is the hidden size of LSTM defined by users. A multi-layer perceptron is constructed to achieve the attention mechanism which adaptive pick the most relevant information in the driving series. The perception accepts the concatenation of h_{t-1}(hidden state in the last time step) and s_{t-1}(cell state in the last time step).

$$e_t = W_e^1 \tanh W_e^2 [h_{t-1}; s_{t-1}] + W_e^3 x_t$$

where $W_e^1 \in \mathbb{R}^T$, $W_e^2 \in \mathbb{R}^{T \times 2m}$ and $W_e^3 \in \mathbb{R}^{T \times T}$ are weight parameters need to be learned by the model. e_t is the vector of weights measuring the importance of features in the driving series at time t(x_t). the softmax function is applied on it to get a_t, which has the sum 1. After that, the driving series will be multiplied with the attention weight a_t to $\bar{x}_t$. Finally the LSTM will accept $\bar{x}_t$ and h_{t-1} as its input, then update the hidden state at that time, h_t. Both the perceptron and the LSTM can be trained together and the encoder focus its attention on some significant feature in the time series.

3.2.2 Decoder

The decoder generate the output y_t from both previous target series $y_1, y_2...y_{t-1}$ and the intermediate vector from encoder's hidden state h. In order to follow the input attention in the encoder, another attention mechanism is applied to pick the relevant information from hidden state of the encoder. Similarly, there is also a multi-layer linear functions to enable the attention. LSTM cell state s_{t-1} and previous decoder hidden states d_{t-1}(with dimension k) are concatenated as input as well.

$$l_t = W_d^1 \tanh W_d^2 [d_{t-1}; s_{t-1}] + W_d^3 h$$

where $W_d^1 \in \mathbb{R}^m$, $W_d^2 \in \mathbb{R}^{m \times 2k}$ and $W_d^3 \in \mathbb{R}^{m \times m}$ are weight parameters need to be learned by the model. l_t represents the importance of the elements of intermediate vector from encoder and β_t is the weight after softmax. An context vector c_t is introduced by summing up the multiplication of $l_1..l_t$ and $h_1...h_t$.

$$c_t = \sum_{i=1}^{T} h_i \beta_i$$

After that the context vector can be concatenated with the history target series $[y_1...y_{t-1}]$ and then it as well as previous decoder hidden state d_t are put into LSTM to update d_t. Finally several linear layers is constructed to fully connect the concatenation of d_T and c_T into the final result.

$$y_t = W_y^1 (W_y^2 [d_T; c_T] + b_1) + b_2$$

4 Implementation

This section will briefly introduce the design. The detailed implementation can be found in the source code.

4.1 Framework

The two neural network models in the project are implemented with a Python deep learning framework called *Pytorch*, which is easy for defining custom deep learning models. In addition, *Pandas*, *Numpy* and *Scikt-learn* is used for data processing, *Matplotlib* for visualize the train and test result.

4.2 Structure

The code structure is simple. It includes a dataset class to do preprocessing of training and testing data, a model class contains the definition of the model introduced before and a trainer class conducting the training and testing. The trainer allows a mini-batch training which can utilizes GPU computation as much as possible.

5 Test and Evaluation

5.1 Environment

Because my laptop because does not have a NVIDIA GPU enabling CUDA, a Google compute engine(GCE) with Ubuntu 16.04 LTS is used for training the model to accelerate the processing speed. The testing phase is much less computation than training so it will be run on my Macbook.

5.2 Datasets

For the first simple model, this project use the OHLCV of AAPL.US as input and target time series.

For DA-LSTM, the stock price of Microsoft is used as driving series and the stock price of Apple is the target series. Since Apple went public earlier than Microsoft, the data is cropped as the same time span. The dataset is divided into training set and validation

set at the split ratio of 0.8.

5.3 Configuration

For each model, it will be trained for 50, 100, 200, 300, 400, 500 epochs. They uses means square error as loss function. The batch size is set as 128, for both training and testing. The model will be automatically saved at each checkpoint.

5.4 Results

5.4.1 DA-LSTM

The prorgame will pick the most two accurate model on testing set and the prediction result given by DA-LSTM is shown in the following figures. Model with 200 epochs and 300 epochs gives the best result. In order to make the figure more clear, the time span is cropped so that it starts from 2000. The black curve denotes the actual data of stock price of Apple and the red curves denotes the predicted result on training set while the blue one denotes the predicted result on testing set.

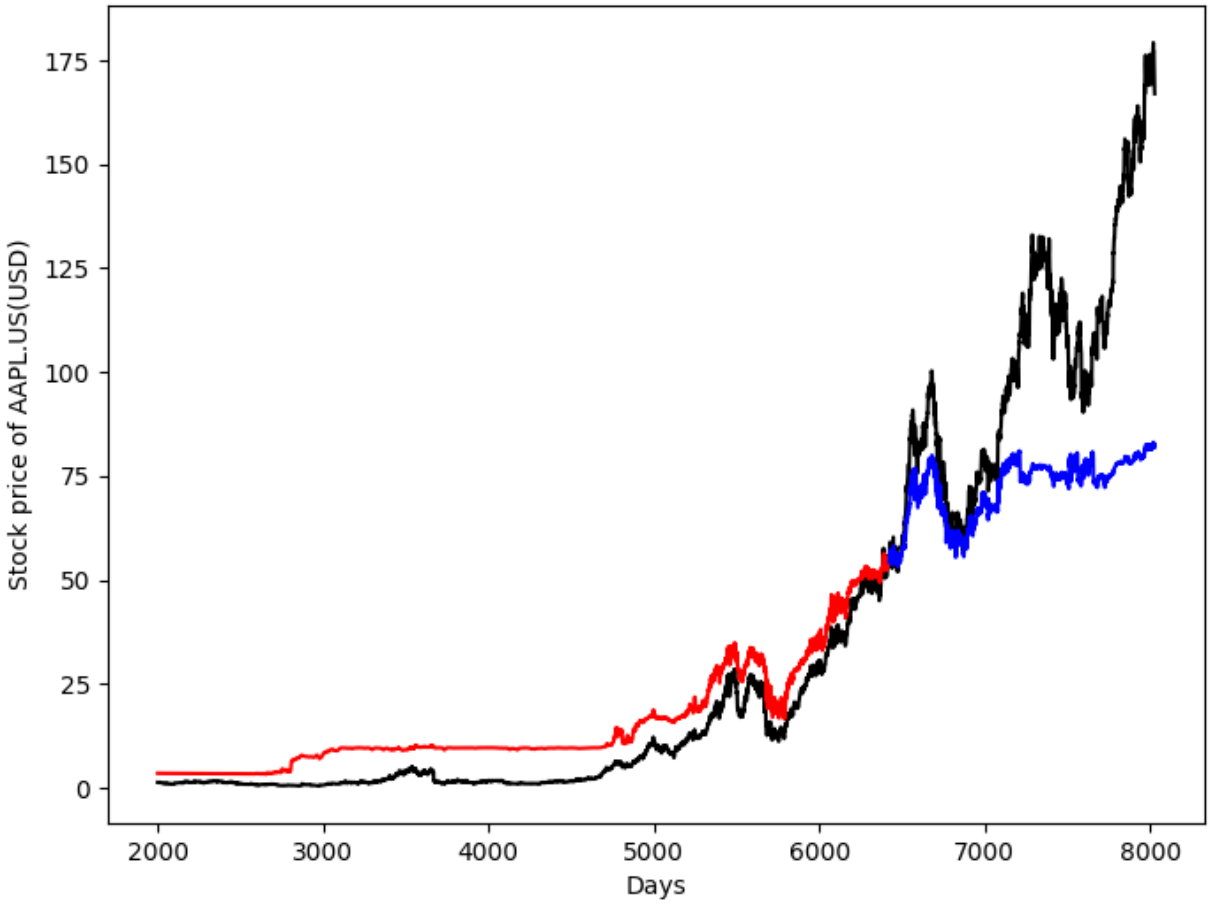

Figure 7: 200 epochs with 128 batchsize - black:ground-truth, red:prediction for training set, blue: prediction for testing set

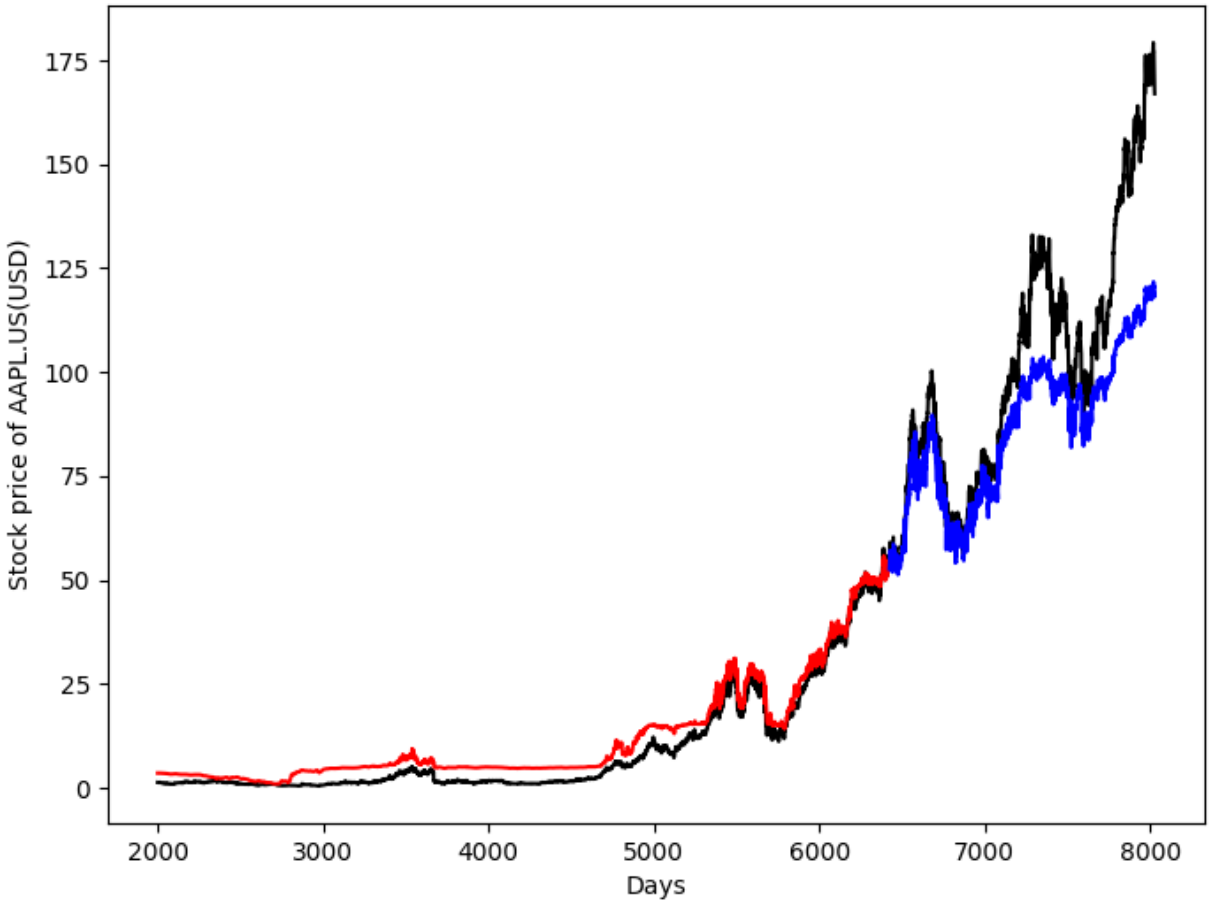

Figure 8: 300 epochs with 128 batchsize - black:ground-truth, red:prediction for training set, blue: prediction for testing set

From Figure 7 and Figure 8 we can see, they both fit the training set quite well, it is same as other models shown in Appendix A. However the difference is, they seem to have an acceptable performance while others are overfitting. The 200-epochs model fits the ground truth at the beginning of testing set but fails later. However it is noticed that the trend of increasing or decreasing is also close to the actual case. The 300-epochs model gives even higher performance since we find the curve is similar to the ground truth.

The predicted price is significantly lower than the truth because the tech stock price

increasing rapidly in the recent years. It is difficult for a model to generalize this case. However from learning how the price going up and down can be useful for financial prediction.

Compared with simple RNN or LSTM model, DA-LSTM definitely gives better predictions due to its two attention mechanism, which makes the encoder learn more useful information from the dataset and the decoder extract the important features from the coding vector.

One more interesting things found is that the model performance varies with the epochs changing. It can be inferred that the model is very parameter sensitive since the results differs much with model hidden size changing.

6 Conclusion

In order to predict the stock price behaviour based on the historic series data, LSTM and encoder-decoder model with attention mechanism are used. DA-LSTM model provides more accurate predicted results compared with simple LSTM because the dual attention mechanism allows the model extract information from both a driving series(can be the price of another stock) and the target series. It can dig out useful features as much as possible so a surprise result is found. However the prediction is very parameter sensitive and now it is difficult to generalize the model on different stocks. This problem can be studied and tried in future.

References

[1] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *ArXiv e-prints* (June 2014).

[2] Mnih, V., Heess, N., Graves, A., and Kavukcuoglu, K. Recurrent Models of Visual Attention. *ArXiv e-prints* (June 2014).

[3] Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., and Cottrell, G. W. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (2017), IJCAI'17, AAAI Press, pp. 2627–2633.

A Full Results

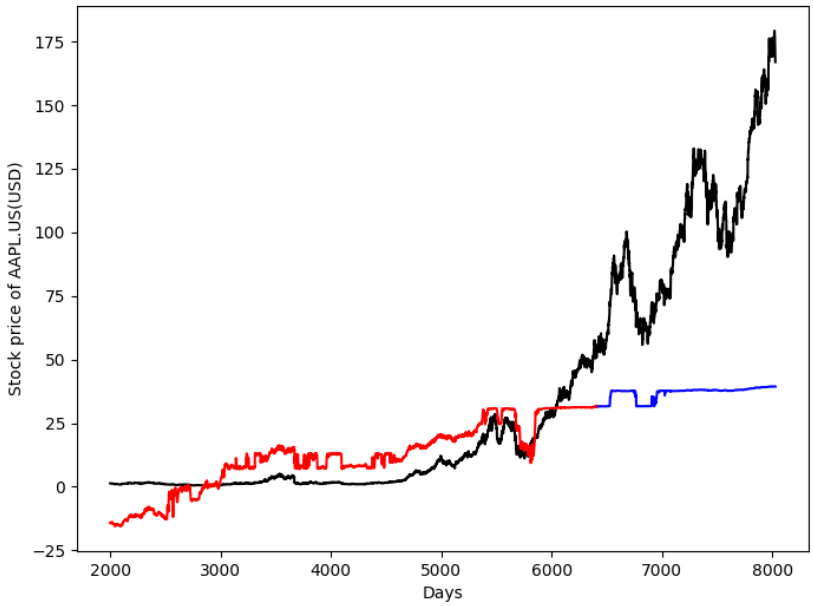

Figure 9: 50 epochs with 128 batchsize

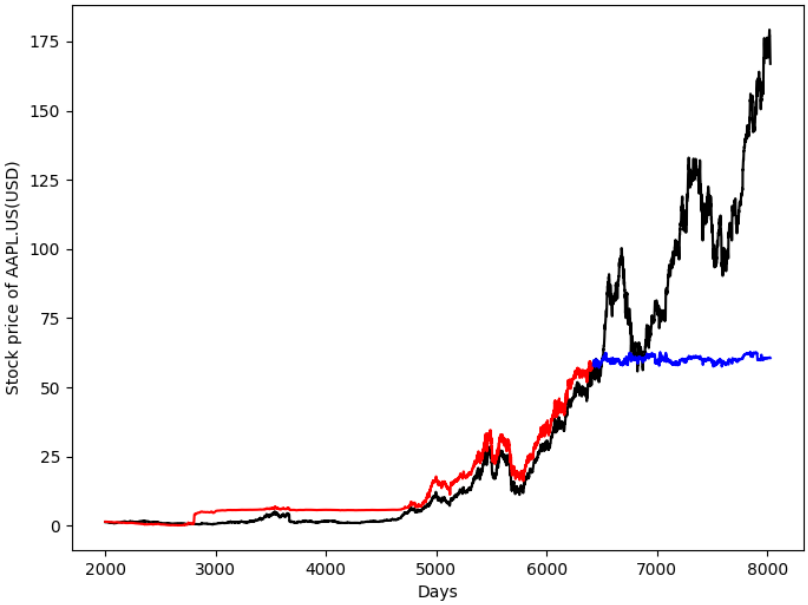

Figure 10: 100 epochs with 128 batchsize

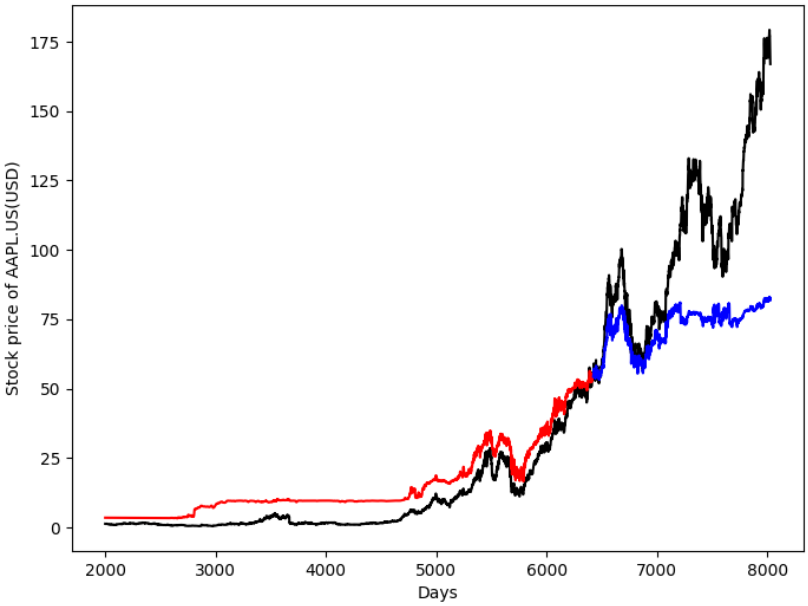

Figure 11: 200 epochs with 128 batchsize

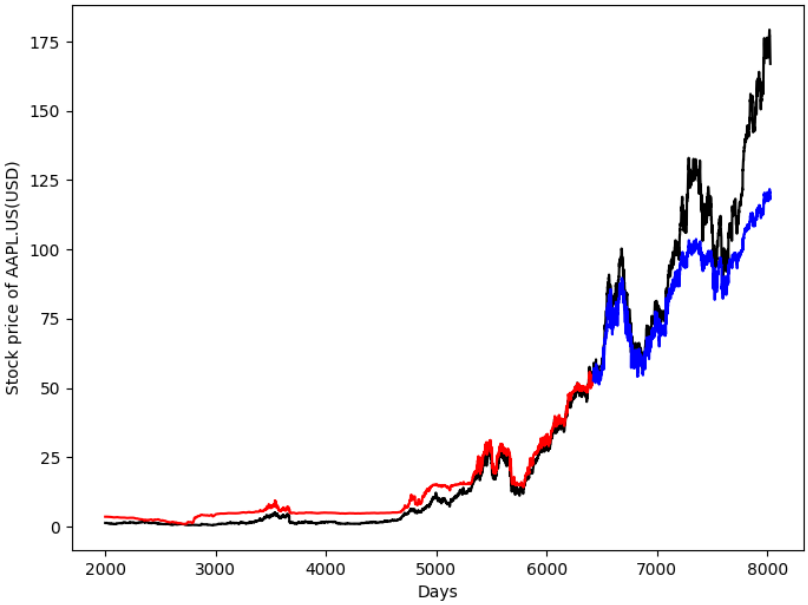

Figure 12: 300 epochs with 128 batchsize

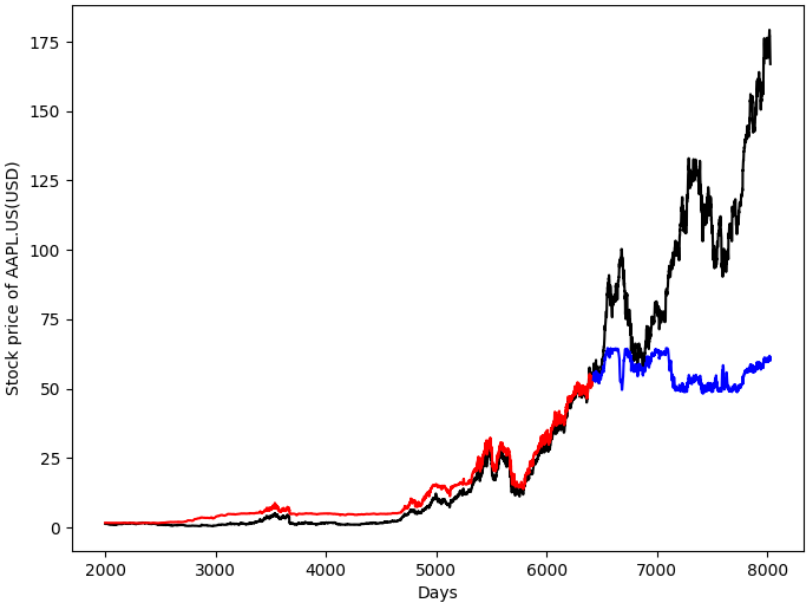

Figure 13: 400 epochs with 128 batchsize

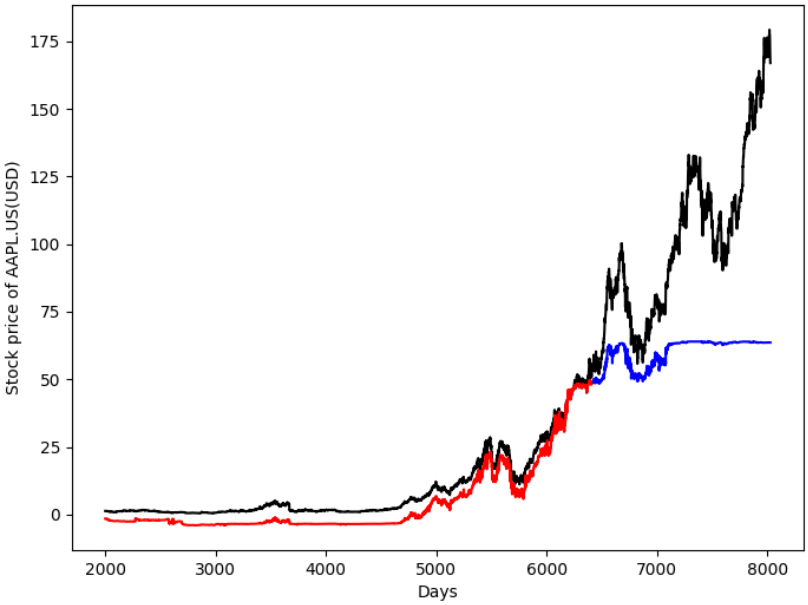

Figure 14: 500 epochs with 128 batchsize

B Minutes

B.1 1st meeting

Date: 26th Feb
Time: 11:30 am
Place: Room 3554
People: Dr.Rossiter, Yu Shengnan
Approval: Approved
Discussion: 1. The framework using. 2. The problem that prediction result shows. 3. Advise on data preprocessing and feature selection.

B.2 2nd meeting

Date: 19th March
Time: 11:30 am
Place: Room 3554
People: Dr.Rossiter, Yu Shengnan
Approval: Approved
Discussion: 1. Show the implementation of new model. 2. Analyze the results 3. Discuss the problem of result visualization 4. Some advice for improving the model

B.3 3rd meeting

Date: 9th April
Time: 11:30 am
Place: Room 3554
People: Dr.Rossiter, Yu Shengnan
Approval: Approved
Discussion: 1. Discuss the strange problem in the results 2. Advice for an automatic model pick implementation

B.4 4th meeting

Date: 9th May
Time: 11:50 am
Place: Room 3554
People: Dr.Rossiter, Yu Shengnan
Approval: Approved
Discussion: 1. Review of the final report 2. Advice for some detail in report structure