Research on the Application of LSTM Model in Predicting Stocks

Jiabo Li[1a]

[a]masteralj@outlook.com

[1]School of Information Engineering, Tianjin University of Science and Technology Beijing, Tianjin, China, 301830

Abstract: As we all know, the price of a stock is the focus of investors' attention and even small changes can cause huge changes in the market. The SSE Composite Index reflects the overall performance of companies listed on the Shanghai Stock Exchange. This experiment investigates in using the LSTM model to predict which variable parameter in the SSE index has the most influence on the results predicted by the neural network. The data set was obtained from the Shanghai Stock Exchange and it was concluded that for the same LSTM model of the neural network, the opening price parameter of the SSE index had the greatest impact on the prediction of the model.

Keywords: LSTM model, Recurrent Neural Network RNN, Stock

1 Introduction

The SSE Composite Index is a composite stock price index based on all stocks listed on the Shanghai Stock Exchange, weighted by issue volume. By analyzing the SSE Composite Index, one can understand the liquidity of the market by understanding the trading volume of the Shanghai market. General RNN models are weak in portraying long memory time series data [1], and RNN training becomes very difficult when the time series are too long. However, the LSTM model solves the problem that RNN models cannot portray the long memory of time series. This experimental study compares the experimental data to find out which parameter has the greatest impact on the prediction accuracy when using the LSTM model to predict the SSE Composite Index.

2 Main technologies

2.1 RNN model

The formula is as follows:

As shown in Equation (1), (2). f is the nonlinear activation function. When calculating S_0, that is, the hidden layer state of the first word, S_{t-1} needs to be used, but it does not exist, so the 0 vector is generally used.

$$O_t = g(V * S_t) \tag{1}$$

$$S_t = f(U * X_t + W * S_{t-1}) \tag{2}$$

As shown in Figure 1. For a traditional RNN (recurrent neural network), it has many w's but they have the same value, and as the data passes through the same unit, the memory of the input is recorded and another input to be predicted is added [2], so the prediction contains all the previous memories plus this input. The weights will cause the error to change in the case of greater than 1, there will be error amplification in the backpropagation, leading to gradient explosion; less than 1, the error will keep shrinking [3], leading to gradient disappearance, thus making the network's weights take a long time to be updated, not allowing the RNN to record all the memories, making the RNN too forgetful.

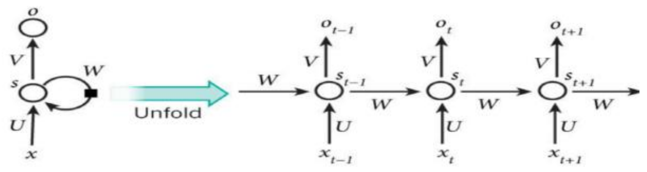


Figure 1 Single layer network architecture

2.2 LSTM model

The formula is as follows:

forget gate: As shown in Equation (3), Choose to forget certain information from the past

$$f_t = \sigma\left(W_f * [h_{t-1}, x_t] + b_f\right) \tag{3}$$

input gate: As shown in Equation (4), (5). Remembering certain information in the present

$$i_t = \sigma\left(W_i * [h_{t-1}, x_t] + b_i\right) \tag{4}$$

$$\tilde{C}_t = \tanh\left(W_C * [h_{t-1}, x_t] + b_C\right) \tag{5}$$

Merging past and present memories: As shown in Equation (6)

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_{\mathrm{t}} \tag{6}$$

output gate: Output, As shown in Equation (7), (8)

$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t * \tanh(C_t) \tag{8}$$

As shown in Figure 2. The long and short term memory model is a special RNN model, which is designed to solve the gradient disappearance and gradient explosion problems during backpropagation, and solves the long memory problem that the RNN model does not have by introducing the gate mechanism. Specifically, the neurons of the LSTM model consist of cells and gates [4]. The cell is the key component of the LSTM model, similar to memory, which is used to store the model's memory. The cell states are subject to change over time, and the saved information is managed by the gate mechanism [5]. The gate mechanism is used to filter a portion of the information, which is implemented by a sigmoid function and a dot product operation. Sigmoid takes values between 0 and 1, while different dot product operations determine the amount of information that passes through the gate. The sigmoid has two values of 0 or 1 representing the abandonment of the message and the selection of full transmission (i.e., full memory), respectively. The LSTM protects and controls the unit states through gates, which are the abandonment gate, the update gate, and the resultant output gate [6]. The unit state is like a transmission belt. It will go along the direction of motion of the transmission belt, where there is only little linear interaction. This approach will make the information flow on the transmission belt.

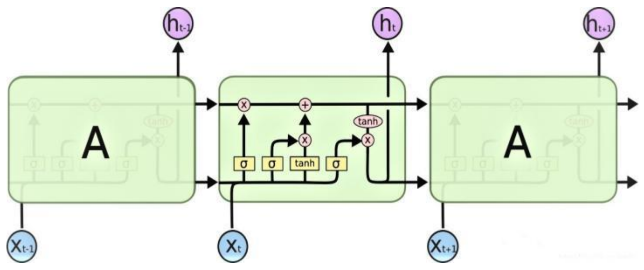


Figure 2 Single layer model

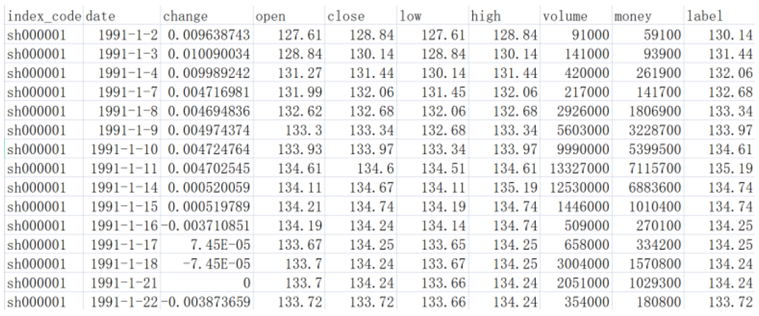
3 Method of realizing model

3.1 Visualize the data

It is necessary to import the dataset first and then format it with some parameters (e.g. date) for later analysis. The dataset comes from the Shanghai Stock Exchange from 1991 to 2015 for the SSE Composite Index. Some of the data sets are shown in Figure 3 below. The stock parameters are shown in the following table 1.

Table 1. Dataset Properties

Stock Parameters	Data volume
opening price	6100
closing price	6100
gain/loss	6100
high value	6100
low value	6100
volume	6100
amount traded	6100



index_code	date	change	open	close	low	high	volume	money	label
sh000001	1991-1-2	0.009638743	127.61	128.84	127.61	128.84	91000	59100	130.14
sh000001	1991-1-3	0.010090034	128.84	130.14	128.84	130.14	141000	93900	131.44
sh000001	1991-1-4	0.009989242	131.27	131.44	130.14	131.44	420000	261900	132.06
sh000001	1991-1-7	0.004716981	131.99	132.06	131.45	132.06	217000	141700	132.68
sh000001	1991-1-8	0.004694836	132.62	132.68	132.06	132.68	2926000	1806900	133.34
sh000001	1991-1-9	0.004974374	133.3	133.34	132.68	133.34	5603000	3228700	133.97
sh000001	1991-1-10	0.004724764	133.93	133.97	133.34	133.97	9990000	5399500	134.61
sh000001	1991-1-11	0.004702545	134.61	134.6	134.51	134.61	13327000	7115700	135.19
sh000001	1991-1-14	0.000520059	134.11	134.67	134.11	135.19	12530000	6883600	134.74
sh000001	1991-1-15	0.000519789	134.21	134.74	134.19	134.74	1446000	1010400	134.74
sh000001	1991-1-16	-0.003710851	134.19	134.24	134.14	134.74	509000	270100	134.25
sh000001	1991-1-17	7.45E-05	133.67	134.25	133.65	134.25	658000	334200	134.25
sh000001	1991-1-18	-7.45E-05	133.7	134.24	133.67	134.25	3004000	1570800	134.24
sh000001	1991-1-21	0	133.7	134.24	133.66	134.24	2051000	1029300	134.24
sh000001	1991-1-22	-0.003873659	133.72	133.72	133.66	134.24	354000	180800	133.72

Figure 3 Part of a data set

3.2 Create the dataset

In this experiment, a specific dataset needs to be created for training, validation and testing before creating the neural network. Considering the real training environment [7], the number of training samples per batch (batch_size), time (time_step) and the number of training sets (train_begin, train_end) are set as parameters here to make the training more mobile.

3.3 Normalize data

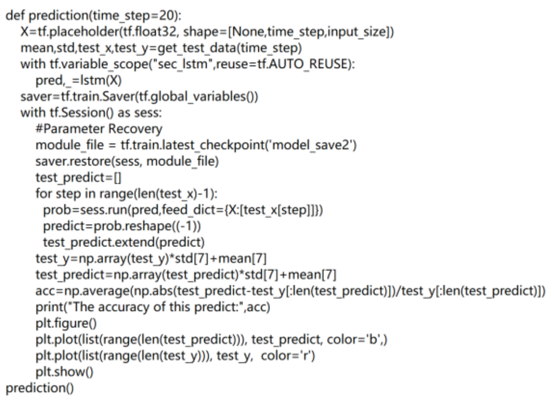
The dataset for this experiment needs to be scaled between 0 and 1. In fact, in order to use our data with a neural network, this experiment needs to normalize its own data. Moreover, this will allow the network to understand the price differences. In fact, there is a big difference between the value of a stock on the first day and the value of today [8]. By normalizing, we reduce the gap and our model can be more accurate. In this paper, we use Z-Score normalization with the formula z=(x-μ)/σ; its main purpose is to unify data of different magnitudes into the same magnitude, which is uniformly measured by the calculated Z-Score value, ensuring comparability among data. The disadvantage of standardization is that, first of all, estimating Z-Score requires the overall mean and variance, [9] which is difficult to obtain in real analysis and mining, and in most cases the mean and standard deviation of the sample are used instead; the results of Z-Score can only be used to compare the results between data, and the real meaning of the data needs to be restored to the original value.

3.4 Loss function

The loss function is a function used to represent the magnitude of the difference between the predicted value X and the true value Y of the network model. It is a non-negative real-valued function, usually denoted by L (Y, f(x)). The loss function is negatively related to the robustness of the model, and the model becomes robust as the loss function decreases [10]. The loss function is a central part of the calculation of risk, and the combination allows the loss function to become a structural risk function. The very commonly used mean squared error loss is taken in the experiments.

3.5 Build neural network training model

First, we define the neural network variables, input layer, output layer weights, bias, and dropout parameters. Then we train the model in the training set, test it, and finally make predictions. The prediction model implementation is shown in the figure 4 below.



```
def prediction(time_step=20):
    X=tf.placeholder(tf.float32, shape=[None,time_step,input_size])
    mean,std,test_x,test_y=get_test_data(time_step)
    with tf.variable_scope("sec_lstm",reuse=tf.AUTO_REUSE):
        pred,_=lstm(X)
    saver=tf.train.Saver(tf.global_variables())
    with tf.Session() as sess:
        #Parameter Recovery
        module_file = tf.train.latest_checkpoint('model_save2')
        saver.restore(sess, module_file)
        test_predict=[]
        for step in range(len(test_x)-1):
          prob=sess.run(pred,feed_dict={X:[test_x[step]]})
          predict=prob.reshape((-1))
          test_predict.extend(predict)
        test_y=np.array(test_y)*std[7]+mean[7]
        test_predict=np.array(test_predict)*std[7]+mean[7]
        acc=np.average(np.abs(test_predict-test_y[:len(test_predict)])/test_y[:len(test_predict)])
        print("The accuracy of this predict:",acc)
        plt.figure()
        plt.plot(list(range(len(test_predict))), test_predict, color='b',)
        plt.plot(list(range(len(test_y))), test_y,  color='r')
        plt.show()
prediction()
```

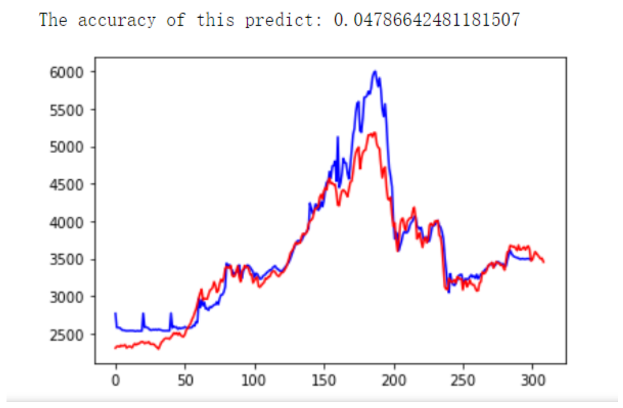
Figure 4 Predictive Models

3.6 Set up a comparison experiment

In this experiment, the dataset has seven parameters used for model training, and in order to find out which parameter has the most influence on the prediction results of the LSTM model, eight experiments are set up, and the first seven times remove a different parameter from the seven parameters each time, and use the remaining six parameters for model training prediction, and the last time use eight parameters to train the model for prediction, and according to the prediction deviation values in the results, it can be deduced which parameter is more important for predicting the SSE Composite Index.

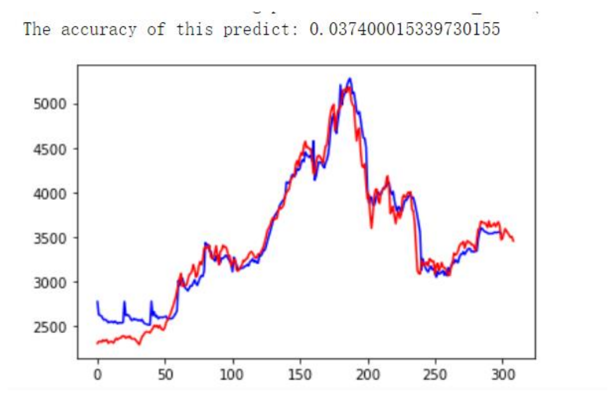
4 Results and analysis

Red represents the predicted value, blue represents the true value.



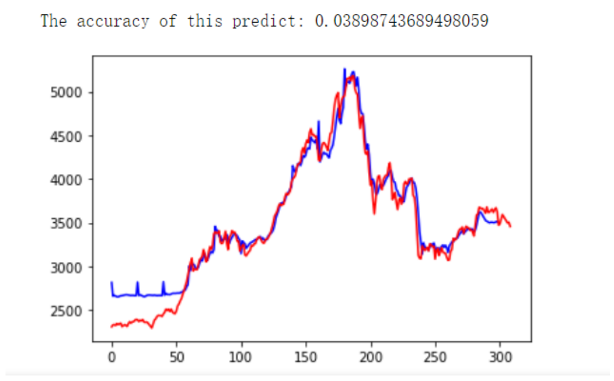
```
The accuracy of this predict: 0.04786642481181507
```

Figure 5 Model prediction results after removing the opening price parameter



```
The accuracy of this predict: 0.037400015339730155
```

Figure 6 Model prediction results after removing the closing price parameter



```
The accuracy of this predict: 0.03898743689498059
```

Figure 7 Model prediction results after removal of the Up and down parameters

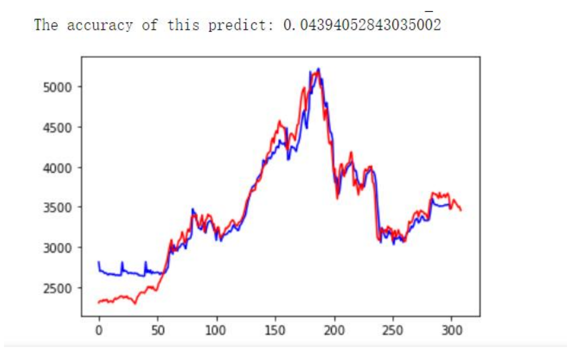

```
The accuracy of this predict: 0.04394052843035002
```

Figure 8 Model prediction results after removing the maximum price parameter

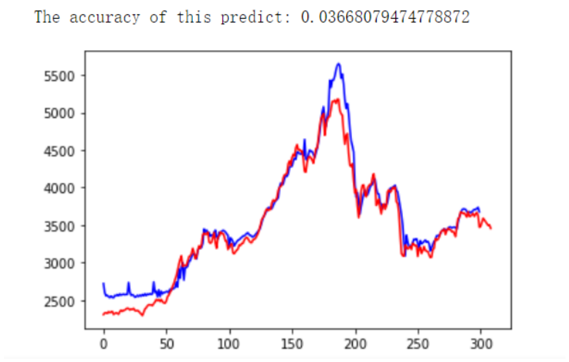

```
The accuracy of this predict: 0.03668079474778872
```

Figure 9 Model prediction results after removing the transaction volume parameter

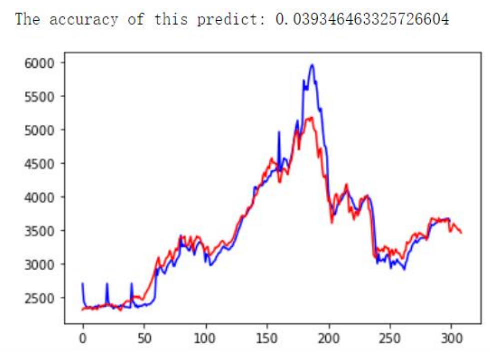
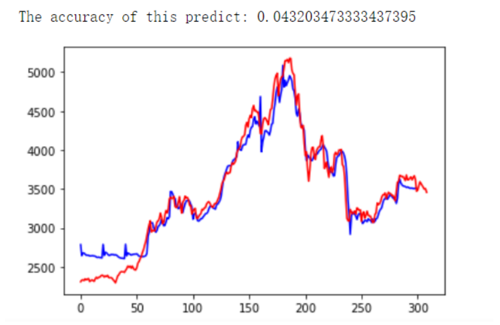
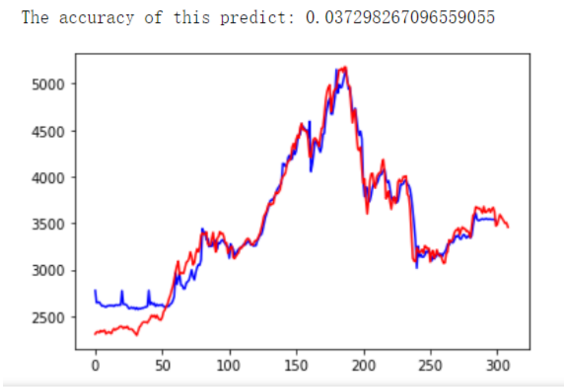

```
The accuracy of this predict: 0.039346463325726604
```

Figure 10 Model prediction results after removing the transaction amount parameter



```
The accuracy of this predict: 0.043203473333437395
```

Figure 11 Model prediction results after removing the up and down parameters



```
The accuracy of this predict: 0.037298267096559055
```

Figure 12 All parameters for model prediction results

Table 2. Model Results

Experiment	Remove one parameter	Deviation value
Experiment 1	opening price	0.0478
Experiment 2	closing price	0.0374
Experiment 3	gain/loss	0.0389
Experiment 4	high value	0.0439
Experiment 5	low value	0.0367
Experiment 6	volume	0.0393
Experiment 7	amount traded	0.0432
Experiment 8	Nothing	0.0372

Each graph from Figure 5 to Figure 11 removes a different parameter predicted by the model. In each graph, the deviation of the predicted value from the true value is calculated. The deviation value is inversely proportional to the accuracy of the prediction. The larger the deviation, the more influence the prediction receives and the worse the curve fit. The smaller the deviation, the better the curve fit. The deviation values calculated from the experiments in Figures 5 to 11 were calculated separately from the deviation values in Figure 12 to find out the changes in the deviation values, and it was found that the deviation values in Figure 5 differed the most from those in Figure 12, where the opening price parameter was removed to forecast the stock. This indicates that the opening price parameter has the greatest impact on the prediction results when using the LSTM model to predict the SSE Composite Index. The SSE Composite Index is a composite stock price index based on all stocks listed on the Shanghai Stock Exchange, weighted by issue volume. Table 2 shows the experimental results.

5 Conclusion

In this experiment, it was found that when using the LSTM model to predict the stock, different parameters of the stock itself play different roles in the prediction results of the model, and in this experiment, the opening price parameter has the greatest impact on the experimental results, which means that this parameter has great significance for the accuracy of the prediction of the stock. In the future, the experiment will continue to investigate whether the effect of different parameters on the prediction can be related to the number of iterations of the model.

References

[1] Akhtar Md. Mobin, Zamani Abu Sarwar, Khan Shakir, Shatat Abdallah Saleh Ali & Samdani Faizan (2022). Stock market prediction based on statistical data using machine learning algorithms. Journal of King Saud University - Science, 101940-.

[2] Banik Shouvik, Sharma Nonita, Mangla Monika & S. Shitharth (2022). LSTM based decision support system for swing trading in stock market. Knowledge-Based Systems, 239

[3] Gao, R., Zhang, X., Zhang, H., Zhao, Q., & Wang, Y. (2022). Forecasting the overnight return direction of stock market index combining global market indices: A multiple-branch deep learning approach. Expert Systems with Applications, 116506.

[4] Huang Lixin, Li Wei & Wu Liansheng (2022). Stock dividend and analyst optimistic bias in earnings forecast. International Review of Economics & Finance.

[5] Kumbure, M. M., Lohrmann, C., Luukka, P., & Porras, J. (2022). Machine learning techniques and data for stock market forecasting: a literature review. Expert Systems with Applications, 116659.

[6] Meshram Vedprakash Vasantrao (2022). The cross-section of Indian stock returns: evidence using machine learning. Applied Economics, 54(16), 1814-1828.

[7] Tong Xin, Wang Jingya, Zhang Changlin, Wu Teng & Wang Yu (2022). LS-LSTM-AE: Power load forecasting via Long-Short series features and LSTM-Autoencoder. Energy Reports, 8(S1), 596-603.

[8] U.A. Md. Ehsan Ali (2022). EUR/USD Exchange Rate Prediction Using Machine Learning. International Journal of Mathematical Sciences and Computing, 8(1), 44-48.

[9] Wang Fengrong & Muchenje Linda (2022). Economic policy uncertainty and stock liquidity: The mitigating effect of information disclosure. Research in International Business and Finance, 59

[10] Yu Yong, Si Xiaosheng & Zhang Jianxun (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures.. Neural Computation, 31(7), 1235-1270.