Advances in Computer, Signals and Systems (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2024.080415
ISSN 2371-8838 Vol. 8 Num. 4

Stock Prediction System Based on Bi-directional LSTM Model

Chenhao Wei[*], Hong Rao

Sun Yueqi Honors College, China University of Mining and Technology, Xuzhou, 221116, China
[*]Corresponding author: 08230916@cumt.edu.cn

Keywords: Chinese stock market, Stock prediction, Bidirectional Long Short-Term Memory network, Investment analysis, Deep learning

Abstract: In the rapid development of China's socialist market economy, stock investment has become a focal point of public attention, and the accuracy of stock market predictions is crucial for investors. This study employs a Bidirectional Long Short-Term Memory (Bi-LSTM) network model to conduct an in-depth predictive analysis of a representative Chinese stock. The research initially selected stock data from 2010 to 2020, which underwent meticulous cleaning and normalization processes, and was prepared for model training through feature engineering and data segmentation. Subsequently, a Bi-LSTM model was constructed and trained, and its performance was evaluated using a validation set. Ultimately, the model's predictive capability was comprehensively assessed through key indicators such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R2). The study demonstrates the high application potential of the Bi-LSTM model in stock market prediction.

1. Introduction

In the era of big data, deep learning has become a key technology for solving complex problems and predicting future trends. This is especially true in the financial sector, where accurately forecasting the dynamics of the stock market is crucial for investors and analysts. Stock price prediction has always been one of the most challenging issues in financial engineering and quantitative analysis. It is influenced by a variety of unpredictable factors such as market sentiment, political events, and natural disasters. However, with the advancement of machine learning technology, particularly in time series prediction models, we now have more powerful tools for predicting stock prices.

Past research has seen many scholars attempt to address this issue. For instance, Guan Jian analyzed methods based on RNN-CNN neural networks[1], while Li Junhao explored the possibility of using linear regression models for stock price prediction[2]. These studies have provided us with valuable knowledge and understanding, but they have also exposed the limitations of these traditional models in dealing with nonlinear and non-stationary time series data.

This study aims to overcome these limitations by adopting Long Short-Term Memory networks (LSTM) to improve the accuracy of stock price predictions. LSTM models are widely used in time series prediction tasks due to their advantages in processing and remembering long-term dependencies. By analyzing the historical price data of a representative stock in the Chinese stock market, we have demonstrated the LSTM model's ability to capture stock price fluctuation trends and verified the model's predictive accuracy through a series of performance evaluation indicators.

The stock price prediction method based on Bidirectional LSTM proposed in this paper is illustrated in Figure 1.

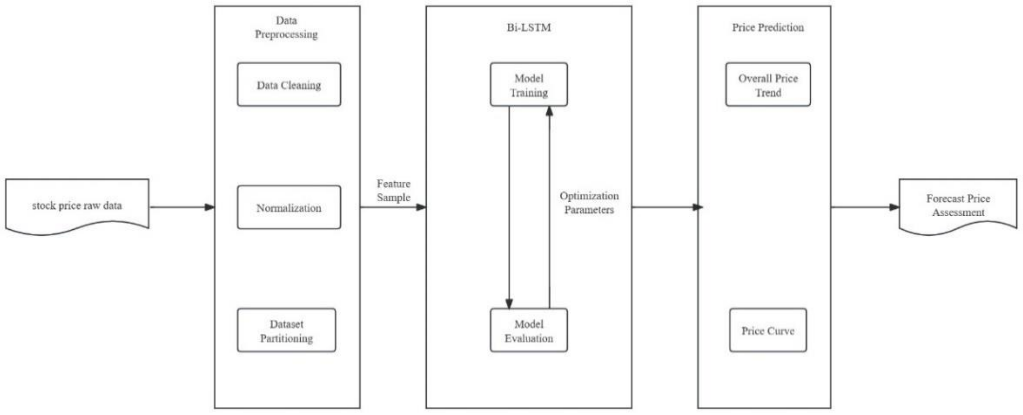

Figure 1: Stock Price Prediction Method Based on Bi-LSTM

2. Literature review

Stock price prediction is a considerably complex issue within the financial domain, exerting significant influence on investment decisions and market analysis. Traditional forecasting methods such as linear regression models and CNN convolutional neural network models, while widely applied in practice, have limitations in dealing with the market's non-linear characteristics and dynamic changes, particularly over long time series.

In recent years, with the advancement of deep learning technology, researchers have begun to explore its potential in stock price prediction. Models capable of handling time series data, such as Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), and Gated Recurrent Units (GRU), have demonstrated superior performance in predicting stock prices.

For example, Sun Chenhao and Wang Lin used an LSTM model to predict the Shanghai Composite Index and compared it with the traditional ARIMA model, showing that the LSTM model performed better on long-term indicators[3]. Additionally, Peng Yan, Liu Yuhong, and Zhang Rongfen demonstrated the practical application value of the LSTM model in actual trading by constructing an LSTM-based trading strategy[4]. Qiao Zhongxue's research on the Shenzhen Component Index with a Bidirectional LSTM model indicated that the Bidirectional LSTM model has better prediction capabilities for time series to a certain extent than the classic LSTM model[5].

In summary, the LSTM model has shown tremendous potential in stock price prediction, but to fully utilize this technology, further research and experimentation are needed. Future work can focus on model optimization, feature engineering, and the integration of other financial theories to improve the accuracy and reliability of predictions.

3. Data Preprocessing

3.1 Data Overview

The data used in this article is derived from a publicly available dataset of a representative stock from the Chinese market. This dataset encompasses a total of 2026 entries spanning a decade from

2010 to 2020, which includes data features as shown in Table 1.

Table 1: Stock Data Feature Table

Data Feature	Meaning
Opening Price	opening price of the day
Closing Price	closing price of the day
Highest Price	highest price of the day
Lowest Price	lowest price of the day
Trading Volume	number of shares traded

3.2 Data Preprocessing Operations

Initially, the raw dataset underwent data cleaning, missing value imputation, and outlier removal.

$$x_{\text{fill}} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

The next step involves normalizing the data, scaling the features to a range of [0, 1].

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{2}$$

Then, the opening prices for the first 2126 days are designated as the training set, and the opening prices for the subsequent 300 days as the test set. Finally, the training and test sets are formatted to suit time series prediction, conforming to the expected input shape for the LSTM model.

4. Model Establishment

4.1 Model Introduction

LSTM is an enhanced type of Recurrent Neural Network (RNN) designed to address the vanishing gradient problem and capture long-term dependencies. Unlike standard RNNs, LSTMs incorporate memory cells and gate mechanisms, which help in selectively retaining or forgetting information, making them suitable for time series data modeling.

4.2 Establishment of the Bi-LSTM Model

This study implements a Bidirectional LSTM model using the TensorFlow framework to explore the deep features of sequence data. The model structure includes LSTM layers in two directions, capable of capturing the dependencies of time series both forward and backward. With the proper configuration of optimizers and loss functions, the model demonstrates good predictive performance after multiple rounds of iterative training.

As illustrated in Figure 2, in the Bi-LSTM, two LSTM layers are arranged in parallel, one processing the forward time steps and the other the backward time steps. These two parallel LSTM unit flows receive input at each time point, and their outputs are merged to form a complete understanding of the current time point. This bidirectional flow allows the network to capture more complex patterns, thereby learning the context information of both past and future simultaneously.

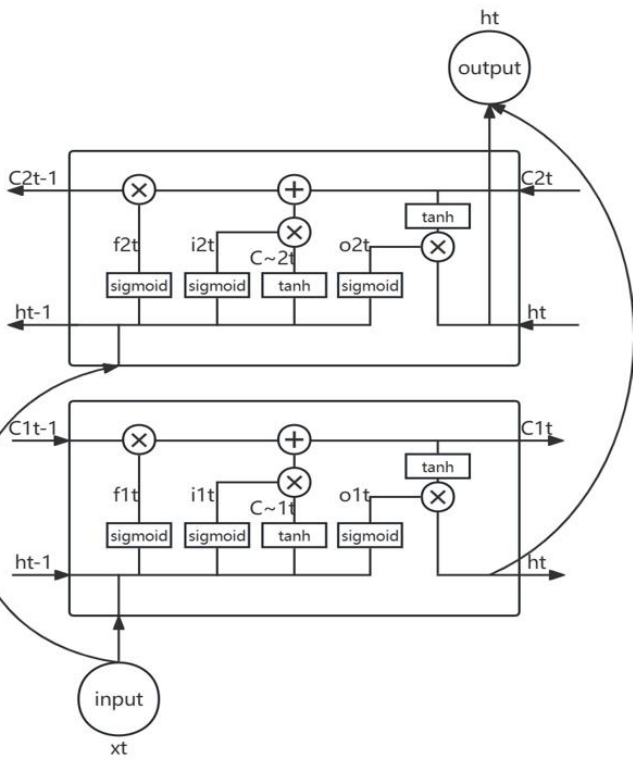

Figure 2: Bi-LSTM Architecture Diagram

Here's a pseudocode representation of the algorithm for constructing the Bi-LSTM model:

```
Step 1. model = Sequential()                    # Define the model structure
Step 2. model.add(Bidirectional(LSTM(units=64, return_sequences=True), input_shape=(time step, Number of features)))          # Add a bidirectional LSTM layer, set the number of cells and the input shape
Step 3. model.add(Dense(Output dimensions)) # Add an output layer and set the output dimensions
Step 4.model.compile(optimizer='adam', loss='Loss function', metrics=['Evaluate metrics']) # Compile the model
Step 5.model.fit(Training data, Training labels, epochs= The number of iterations, batch_size= Batch size)                                        # Train the model
Step 6.model.evaluate(Test data, Test labels) # Evaluate model performance
Step 7.predictions = model.predict(New input data) # Make predictions
```

4.2.1 Model Architecture

Similar to the conventional LSTM, the hierarchical structure of the Bi-LSTM encompasses an input layer, forget gate, input gate, cell state, output gate, and hidden layer. These layers collaborate to enable the Bi-LSTM to effectively process time-series data.

(1) Input Layer: Receives the input of time-series data and conveys this information to the subsequent layer.

(2) Forget Gate: Determines which information to discard from the cell state. This is computed through a sigmoid layer that outputs values between 0 and 1, indicating the extent to which information is retained or discarded.

(3) Input Gate: Decides on the new information to be stored in the cell state. The sigmoid layer determines which values to update, followed by a tanh layer that creates a new candidate values vector.

(4) Cell State: Maintains short-term and long-term memories, updated based on the decisions of the forget gate and input gate.

(5) Output Gate: Determines which part of the cell state to output. The sigmoid layer ascertains which portion of the cell state will be outputted.

(6) Hidden State: Represents the current output at any given moment, calculated based on the cell state and decisions of the output gate.

4.2.2 Parameter Configuration

(1) Timestamp Length: Defined as the sequence length within the data segmentation function, set to 40.

(2) Number of Epochs: The iteration count for model training, set to 20.

(3) Optimizer: The Adam optimizer is employed during model compilation, with a learning rate set at 0.01.

(4) Loss Function: Mean Squared Error function is utilized as the loss function during model compilation.

(5) Batch Size: During model training, the batch size is set to 64.

(6) Validation Frequency: Set to 1, indicating validation occurs after every epoch.

(7) Number of LSTM Units: Configured to 50 units.

(8) Activation Function: Set to the ‘relu’ function.

4.2.3 Training Process

The opening prices for the first 2126 days are used as the training set, with the subsequent 300 days’ opening prices serving as the test set. During each epoch, the model undergoes forward and backward propagation based on the training data to update weights. A validation set is established to monitor model performance and prevent overfitting.

5. Model Resolution

5.1 Selection of Evaluation Metrics

(1) Mean Squared Error (MSE): MSE is the average of the squares of the differences between predicted and actual values, suitable for continuous numerical prediction problems. It is defined mathematically as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

(2) Root Mean Squared Error (RMSE): RMSE is the square root of MSE, measuring the prediction error in the same units as the original data and is sensitive to outliers. It is mathematically expressed as:

$$RMSE = \sqrt{MSE} \tag{4}$$

(3) Mean Absolute Error (MAE): MAE represents the average of the absolute differences between the predicted and actual values. It is less sensitive to outliers and offers a more intuitive measure of prediction accuracy. The formula for MAE is given by:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i | \tag{5}$$

(4) Coefficient of Determination (R^2): The R^2 metric quantifies the extent to which a model accounts for the variability in the data, ranging between 0 and 1. Values closer to 1 indicate a better

fit of the model to the data. It is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{6}$$

5.2 Analysis of Model Resolution

The number of hidden units determines the complexity of the LSTM layer. Fifty hidden units can enhance the model's fitting capability while also ensuring that overfitting is avoided.

The number of iterations dictates the rounds of model training. Too few iterations may lead to underfitting, while too many may cause overfitting. After multiple experiments and employing early stopping, it was determined that twenty training epochs are optimal.

The learning rate controls the magnitude of model parameter updates during each iteration; a learning rate that is too high may result in unstable training, whereas one that is too low can slow down the model's convergence. Figure 3 illustrates the learning loss of the model under the learning rate of 0.1 used in this study.

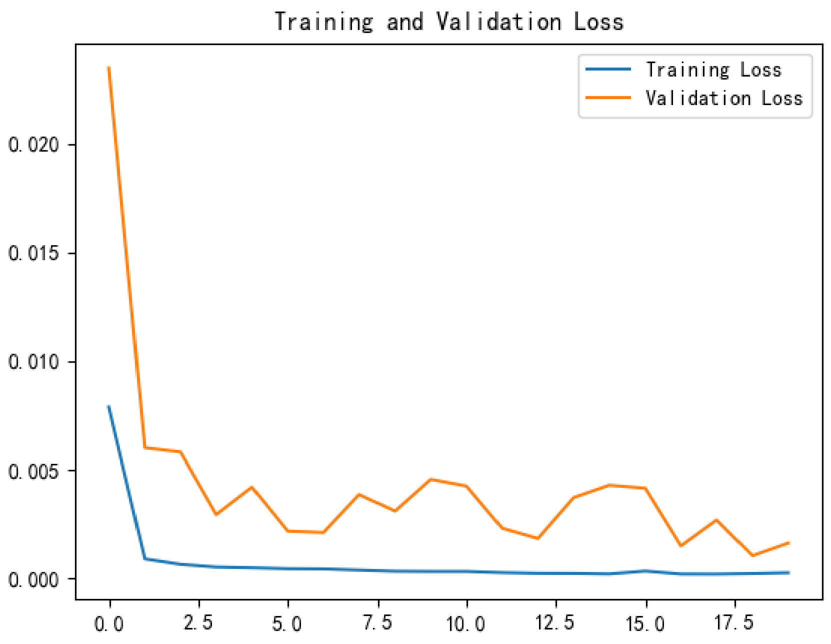

Figure 3: Training and Validation Loss Graph

As depicted in Figure 3, the training loss (represented by the orange line) starts from a higher value and rapidly decreases with the increase in epochs, then gradually stabilizes. This indicates that the model is progressively adapting to the training data.

The validation loss (represented by the blue line) also starts from a higher value and decreases as the epochs progress, but the decline is less pronounced, with slight fluctuations in the later stages. This may suggest that the model has good generalization capabilities on the validation dataset, but there might also be a slight overfitting.

The x-axis represents the epoch, ranging from 0 to 17.5. The y-axis represents the loss value, ranging from 0 to 0.020.

Both lines exhibit higher loss values at the initial stage, but as training progresses, the loss values decrease. The rapid decline and subsequent stabilization of the training loss indicate effective model training.

5.3 Conclusions of the Experimental Section

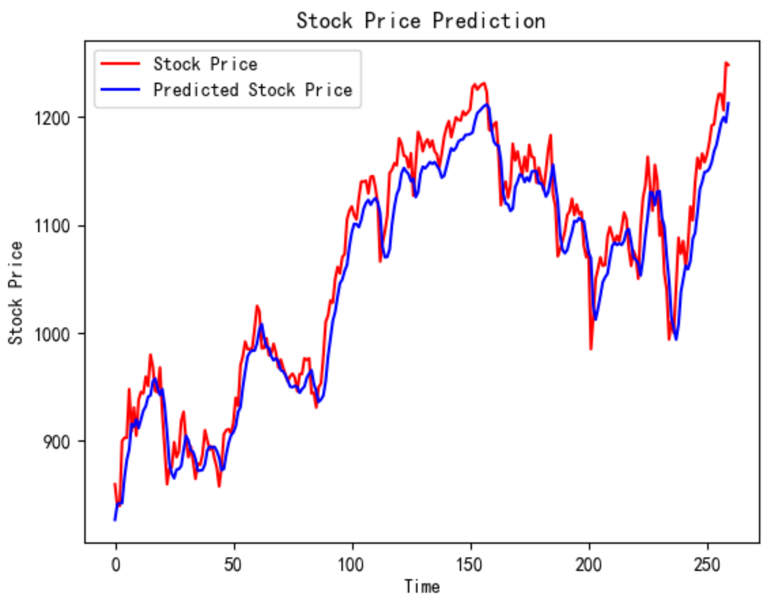

Figure 4: Predicted Price Trend Graph

As depicted in Figure 4, the predictive model is adept at capturing the overall trend of stock prices. Nonetheless, there are instances where notable discrepancies between the predicted and actual values are observed. Such differences may stem from the model's inability to fully capture all the factors that influence stock price fluctuations or due to the stochastic volatility of the stock market.

The experiment yielded the following results:

Table 2: Model Evaluation Metrics Table

MSE	710.1336
RMSE	26.6483
MAE	21.6668
R^2	0.9328

5.4 Conclusions of the Experimental Section

As indicated in Table 2, the values of MSE and RMSE are relatively high, suggesting that the model may have significant errors in certain predictions. However, the R^2 value is close to 1, indicating that the model fits the data well overall. This may mean that the model can make accurate predictions in most cases, but there might be substantial prediction errors in extreme cases or with outliers.

6. Conclusion

In the stock securities trading market, knowing the general trend and price of stocks in the near term can provide investors with considerable reference. This paper, after preprocessing operations such as data cleaning and normalization on typical stock data from the Chinese market, selected a Bidirectional LSTM model and explored suitable numbers of units, iterations, and learning rates through comparative experiments. The study successfully captured the short-term trend of the stock, highly consistent with the actual price trend. However, there are still prediction deviations at some critical points.

Future considerations include further improvements and optimizations in feature engineering, hyperparameter optimization, and interpretability: improving feature extraction methods to integrate more domain knowledge into the model; using automated tools or Bayesian optimization to select the best combination of hyperparameters; and researching how to enhance the interpretability of the LSTM model to make it easier to understand and apply.

References

[1] Guan Jian. Research on Stock Price Prediction Method Based on RNN-CNN Model[D]. Nanjing University of Information Science & Technology, 2023.
[2] Li Junhao. Stock Price Prediction Model Based on Improved Multiple Linear Regression[J]. Science Technology and Economy Market, 2019, (08): 61-62+64.
[3] Sun Chenhao, Wang Lin. Prediction and Analysis of Shanghai Composite Index Based on ARIMA and LSTM[J]. Information and Computer (Theoretical Edition), 2023, 35(02):29-31.
[4] Peng Yan, Liu Yuhong, Zhang Rongfen. Modeling and Analysis of Stock Price Prediction Based on LSTM[J]. Computer Engineering and Applications, 2019, 55(11):209-212.
[5] Qiao Zhongxue. Comparative Study of Shenzhen Component Index Based on Bi-directional LSTM[J]. Industrial Innovation Research, 2020, (14): 83-84.