Stock Price Prediction based on Stacked-LSTM with Attention Mechanism

Haoqi Gu, Junwei Li, Jingyi Li, Fengxu Tu
{haoqigu, jlyc8, ljy668, fengxutu}@bu.edu

1. Task

Stock price prediction has always been a hot but challenging task mainly due to the unprecedented changes in economic trends on one hand and randomness in the stock market on the other hand. Researchers usually derive a large number of factors from raw data, such as historical prices, textual data from social media and corporate profits. However, during the training process, more factors not only increase computation complexity but sometimes even be harmful to the prediction performance. Thus, in our project, only use historical stock price as a training factor to fit different models. Additionally, In the use of various forms of prediction approaches , particularly regression analysis, it is necessary to assess the accuracy of the predictions because they have some limitations in their application.

The main objective of this article is to investigate which forecasting methods can be the best prediction which respect lower forecast errors and higher accuracy of forecasts.

Varieties of stochastic models were proposed in the past decades. The most known method is univariate Auto-Regressive Moving Average (ARMA) for a single time series data. The ARMA consists of Auto-Regressive(AR) and Moving Average (MA) models. Auto-Regressive Integrated Moving Average (ARIMA) is a special type of ARMA with differencing in the model. In addition, there are various machine learning and deep learning based algorithms that have gotten attention to forecast non-linearity and complexity in time series data in recent years including Support Vector Machine (SVM), Random Forests (RF), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM). In particular, LSTM has been used in finance data to do time-series prediction and it achieved good performance.

In this paper, Stacked-LSTM, and Stacked-LSTM with attention models are compared in their performance and accuracy. LSTM approach is used due to its use in long period series-time data prediction. Attention-LSTM was chosen because of its ability to assign different weights to different factors. chosen

2. Related Work

2.1 Stock price prediction

Forecasting time series data is an important and difficult subject in economic, business and financial. Traditionally, there are several approaches effectively predicting the future series data through Autoregressive (AR), univariate Moving Average (MA), and Autoregressive Integrated Moving Average (ARIMA). The ARIMA model has demonstrated better performance in precision and accuracy of forecasting time series data[1].

2.2 ARIMA Model

Many researchers and investors use the ARIMA model ,which is generalized by Box and Jenkins[5], to forecast future stock price by historical data. For a long time, ARIMA has been a popular approach in the field of time series pridiction. Although ARIMA models are very universal in Economic and financial forecasts [6], they have some major limitations [7]. For example, it is difficult to build a model for the non-linear relationships between variables in a simple ARIMA mode. In addition, There is an assumption that a constant standard deviation exists In the ARIMA model. The assumption may be not satisfied with practical problems.

2.3 RNN Model

In recent years, the novel approach Recurrent Neural Network (RNN) , which is a class of artificial neural networks, was proposed. Unlike ARIMA, RNN models are designed not only for time series data, but also for majority sequence data. RNN can use their internal state memory to process sequences of inputs. In the RNN model, each sample can be assumed to be dependent on previous ones. In fact, the idea behind RNN is to predict future trends by learning previous states in sequential datasets. Thus, RNN has feedback loops in the recurrent layer, which can maintain information from the previous stage. However, due to Gradient vanishing and exploding problems, RNN can only remember recent contents [2]. Therefore, Long Short-Term Memory (LSTM) networks, a modified version of RNN, are designed and extremely popular in stock market prediction. In addition, LSTM makes it easier to remember past data in memory[3].

2.4 Attention Mechanism

The attention mechanism is one of the most valuable breakthroughs in Deep Learning research in the last decade. It has spawned the rise of so many recent breakthroughs in natural language processing (NLP) [4], including the Transformer architecture and Google's BERT. It is, to some extent, motivated by how we pay visual attention to different regions of an image or correlate words in one sentence.

2.5 LSTM Model

Long Short-Term Memory (LSTM) is a special case of Recurrent Neural Network (RNN) with the capability of remembering the value from earlier stages for the purpose of future use. LSTM is a bunch of cells. Each cell consists of an input gate, forget gate, output gate. Important information in a sequential data stream can be captured and stored by gates deposing, filtering, or storing to the next cell. Thus, the gates which are based on a sigmoidal neural network layer, enable the cell to let data pass through or dispose. The sigmoidal layer can map values in a certain range of zero to one, which like a 'Valve' to control how much data passes through the cell. More precisely, zero value implies nothing passes through; whereas, one means everything passes through.

3. Approach

The method proposed in [1] is to use a conventional 2 stacked layers LSTM model to predict stock price. The main idea of RNN is to apply the sequential observations learned from the earlier stages to forecast future trends. We use LSTM to overcome the drawback of RNN in capturing long term influences. LSTM introduces the memory cell that enables long-term dependency between time lags. The memory cells replace the hidden layer neurons in the RNN and filter the information through the gate structure to maintain and update the state of memory cells. The gate structure includes input gate, forget gate and output gate.

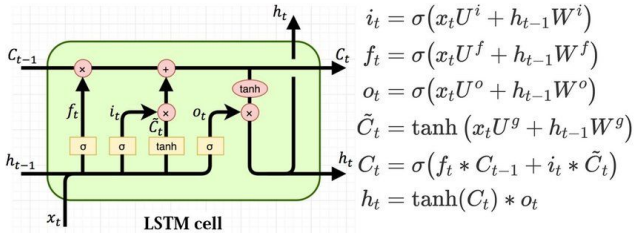

Figure 1. LSTM cell

As Figure 1 shows, the forget gate determines which cell state information is discarded from the model. Its main function is to record the amount of information reserved from the previous cell state to the current cell state. The input gate determines how much the current time network input xt is reserved into the new cell state Ct, it avoids feeding the unimportant information into the current memory cell. The output gate controls how much the newly created cell state will be discarded, the output information is firstly determined by a sigmoid layer, then the newly created cell state is processed by tanh, together with the sigmoid output to determine the final output.

In our project, we implemented the stacked LSTM based on the idea in [5] to forecast the stock price. Due to the higher stochasticity of financial time series, we built up a stacked LSTM model and a stacked LSTM with attention layers respectively, and to evaluate their performances. We expected one of those two models, the stacked LSTM model and model with an attention layer, can capture more stochasticity within the stock market due to its more complex structure.

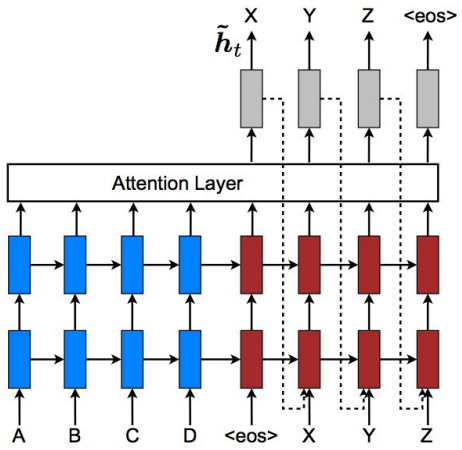

Figure 2. LSTM with Attention Layer

Figure 2 shows the architecture of LSTM with an attention layer. The input from A to Z represents the input of a LSTM cell, which in our project is the stock information in a certain length of period where we try to learn trends from right before the predicted period.

We first used the RNN module to learn hidden states from input data. And input the hidden states to the attention layer to calculate the relation vector e. And we did softmax normalization to obtain the distribution of attention a.

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T} exp(e_{tk})}$$

Intuitively, the attention weights represent the relationship between the stock prices of target day and stock prices of input daty. After compute a, we put the outcome through a linear layer to obtain the desired feature of predicted data, which in our project is 2 (open and close price).
By computing the loss and backprop the network, our module learned the trends between the input data and future stock prices.

In this project, we used some Python libraries to realize the data proprocess, visualization, model training and testing. The libraries we will use include, but not limited to, Pytorch, Pandas, Matplotlibs and Numpy.

4. Dataset

In our project, we use the SSE Composite(Shanghai securities composite index), Apple Inc (AAPL), Amazon Inc(AMZN), Facebook Inc (FB) dataset to train our different models. With fixed time windows T=20 and stride 1 we generated numbers of features and labels. In the experiment, those datasets were splitted as 70% samples for training set, 15% for validate set, 15% for testing set respectively.

4.1 Data

This dataset contains historical daily information as follows:

SHA: Dec 20, 1990 to Dec 10, 2015.
APPL: Dec 12, 1980 to Apr 01, 2020
FB: May 18, 2012 to Apr 01, 2020
AMZN: May 05, 1997 to Apr 01, 2020

This dataset has 7 features. And we only use the feature 2 to 7 to train.

1) Date: specifies trading date
2) Open: opening price
3) High: maximum price during the day
4) Low: minimum price during the day
5) Close: close price adjusted for splits
6) Adj Close: adjusted close price adjusted for both dividends and splits.
7) Volume: the number of shares that changed hands during a given day

5. Results

In our project, we build three different models based on stacked-LSTM and stacked-Attention-LSTM, and we compare the performance among three models for each dataset.

5.1 Metrics

This project uses 4 different metrics namely RMSE (Root Mean Square Error), MBE (Mean Bias Error), MAPE (Mean Absolute Percentage Error), and RF Accuracy (Rise and Fall Accuracy) to evaluate the performance of models. And we use real stock price which is not normalized and denormalized prediction from the model as the input of metrics. We define the metrics RMSE, MBE, MAPE, and RF Accuracy as follows, where y_j is the real stock price, $\hat{y_j}$ is the denormalized prediction price.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\widehat{y_j} - y_j\right)^2}$$

$$\mathrm{MAPE} = \frac{1}{N}\sum_{j=1}^{N}\left|\frac{\widehat{y_j} - y_j}{y_j}\right| \times 100\%$$

$$\mathrm{MBE} = \frac{1}{N}\sum_{j=1}^{N}\left(\widehat{y_j} - y_j\right)$$

$$\text{RF Accuracy} = \frac{1}{N-1}\sum_{j=1}^{N-1}\left(\mathbb{1}\left[\mathrm{Trend}(y_j) = \mathrm{Trend}(\widehat{y_j}) = 1\right] + \mathbb{1}\left[\mathrm{Trend}(y_j) = \mathrm{Trend}(\widehat{y_j}) = -1\right]\right) \times 100\%$$

$$\mathrm{Trend}(y_j) = \begin{cases} 1, & y_j < y_{j+1} \\ -1, & y_j > y_{j+1} \\ 0, & \text{otherwise} \end{cases}$$

$$\forall j = 1, 2, \cdots, N-1$$

RMSE and MBE can show the averaged error of the test data directly, but they may be affected by the magnitude of data itself, which means the value of the real stock price can influence the value of both RMSE and MBE. So we cannot make sure whether the model is good if we only have RMSE and MBE. We decide to use MAPE to evaluate the "relative error" of the model because it can get rid of the influence of the stock price itself.
In this project, we also want to know whether our model can well predict the trends of rise and fall of stock price between every two consecutive days. So we define the RF Accuracy to evaluate the trends predicted by model. The RF Accuracy is defined as the percentage of the number of correct prediction days and the number of total days.
Firstly, we define the concept of "rise" and "fall". Intuitively, we say the stock price is rising if the current price is smaller than the next day price; say the stock price is falling if the current price is larger than the next day price. Therefore, we have the more formal mathematical definition that "rise" if $y_j < y_{j+1}$ and "fall" if $y_j > y_{j+1}$, we only focus on strictly rising and strictly falling. The function $Trend(y_j)$ outputs 1 if "rise", -1 if "fall", 0 otherwise. As for computing the RF Accuracy, we sum up the indicator function, and finally divide by the number of total consecutive days.

5.2 Tuned parameters

- Stacked-LSTM
  - Input dimension: 6
  - Hidden dimension: 100
  - Output dimension: 2
  - Time series length: 20
  - Output time series length: 1
  - LSTM layers: 2
  - Dropout rate: 0.2
  - Batch size: 64
- Optimizer
  - Adam
  - Learning rate: 0.0005
- Training
  - Epoches: 200 (with early stop mechanism)

After validation for different models based on LSTM, stacked-LSTM, and stacked-Attention-LSTM, we find that in most cases the stacked-LSTM model with tuned parameters has the best performance on our datasets.
The input dimension is 6 which includes open price, close price, lowest price, highest price, adjusted close price, and volume of change hands. We only predict open and close prices of one day in this project, so the output dimension is 2 and the output time series length is 1.
From the conclusion of paper [8], we initially set the number of LSTM layer 2, after validation we find that 2 is the best parameter of LSTM layers.
The other parameters are determined after tuning on the validation set.

5.3 Training results

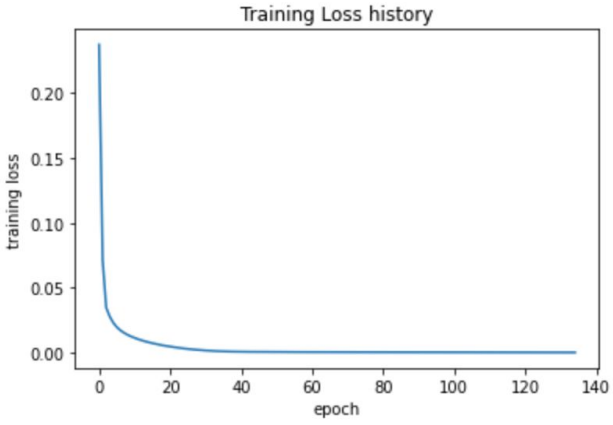

Figure 3. Training loss of stacked-LSTM on SHA dataset

Figure 3 is the training loss of stacked-LSTM with the tuned parameters described previously. The model stops training after 135 epochs because of the early stop mechanism, and the final loss is around 0.0001712. With those parameters, the prediction result is good on our dataset. Figure 4 is the plot of prediction open price and real open price on the validation set of SHA dataset. The prediction curve fits the real price curve well.

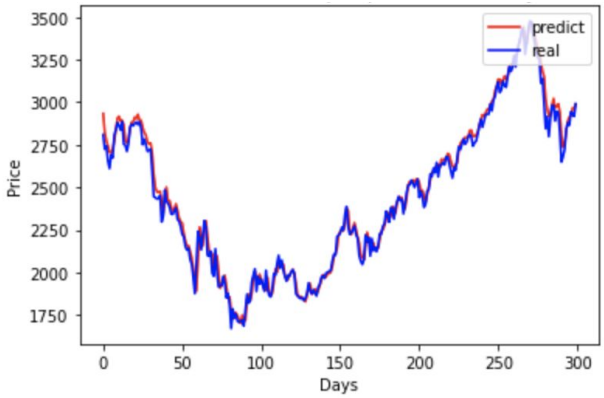

Figure 4. Prediction and real open price on SHA dataset

5.4 Evaluation Results

Metrics \Stocks	AAPL		AMZN		FB	
	open	close	open	close	open	close
RMSE	1.6509	1.3135	7.3241	9.1050	4.2690	4.2661
MBE	0.9322	0.4986	0.6719	-1.2041	2.1717	2.0651
MAPE	1.8718 %	0.6912 %	0.1623 %	0.4505 %	1.7943 %	0.9954 %
RF Accu.	55%	50%	61%	51%	54%	52%

Table 1. Evaluation results of stacked-LSTM model on AAPL, AMZN, FB datasets

From table 1, we can see that the 2 layers stacked-LSTM model trained on AMZN has better performance than the other models on other dataset because it has the smallest MAPE. As for models of AAPL and FB, they have similar MAPE but AAPL has smaller RMSE and MBE, which means the model of AAPL has better performance than the model of FB. We can see that the plots in Figure 5 are consistent with previous analysis.

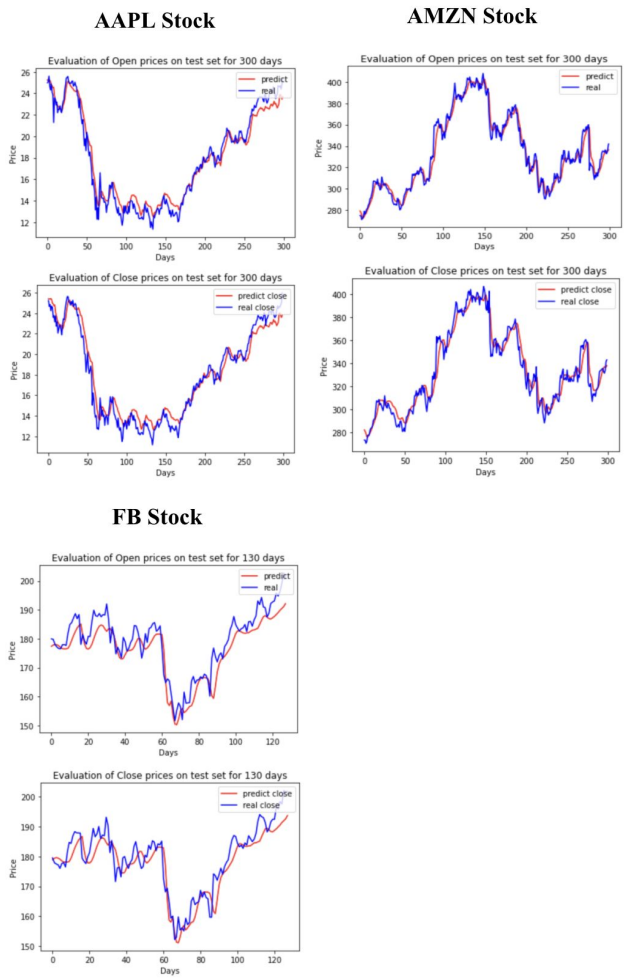

Figure 5. Evaluation results of models trained on AAPL, AMZN, and FB datasets

5.5 Comparison with ARIMA model

Also, we have compared the outcomes with other feasible time series forecasting approaches like ARIMA.

The parameters (p, d, q) of the ARIMA model were determined to be (20, 1, 0) based on the MSE on the training set and the time step we want. This actually makes the model an ARMA model, where no differencing was performed to change the stationarity. And the prediction depends on the stock data of a certain time period which is 20 days.

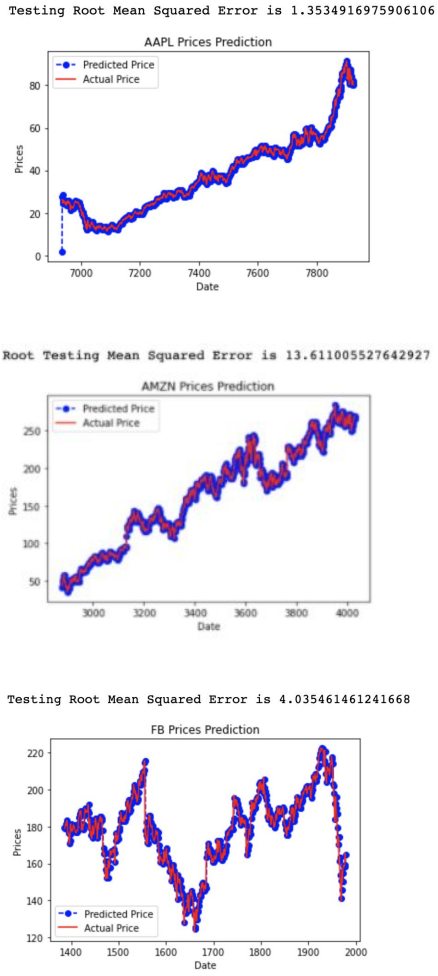

Figure 6. Evaluation results of ARIMA models trained on AAPL, AMZN, and FB datasets

	AAPL	AMZN	FB
RMSE (stacked-LSTM)	1.65	7.32	4.03
RMSE (ARIMA)	1.35	13.6	4.27

Table 2. RMSE obtained from both ARIMA model and Stacked-LSTM model on AAPL, AMZN, FB datasets

Figure 6 shows the outcomes of arima forecasting on 3 stocks. According to table 2, the RMSE obtained from the ARIMA model for those 3 stocks are either similar or worse than our outcomes.

We found several interesting observations on the ARIMA outcome. First, the performance varies with stocks to a great extent. Second, then the test set is getting further and further away from the train set, the accuracy goes down dramatically.

In another word, the ARIMA model gave a linear prediction from test data, though the results suggested the ARIMA model did not perform well in predicting non-linearity and long-term prediction. At least not as good as our model.

6. Detailed Roles

Task	Deadline	Lead
Data preprocessing	10/18/2020	Fengxu Tu, Jingyi Li
Architecture design	11/01/2020	All
Data loader, Net class	11/05/2020	Haoqi Gu, Fengxu Tu
Evaluation	11/07/2020	Jingyi Li, Junwei Li
Project status report	11/11/2020	All
Parameters Tuning	11/19/2020	Junwei Li
Test & Comparing	12/01/2020	Haoqi Gu, Fengxu Tu
Final report	12/06/2020	All
Presentation	12/07/2020	All
Github repo	12/10/2020	Haoqi Gu, Fengxu Tu

7. Code repository

Project Link:
https://github.com/H40Q1/523_stock_predict

References

1) S. Siami-Namini, N. Tavakoli and A. Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, 2018, pp. 1394-1401, doi: 10.1109/ICMLA.2018.00227.
2) Ribeiro, Antônio H, Tiels, Koen, Aguirre, Luis A, and Schön, Thomas B. "Beyond Exploding and Vanishing Gradients: Analysing RNN Training Using Attractors and Smoothness." (2019). Web.
3) Lipton, Zachary C, Berkowitz, John, and Elkan, Charles. "A Critical Review of Recurrent Neural Networks for Sequence Learning." (2015). Web.
4) Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. “Attention Is All You Need.” (2017)
5) G. Box, G. Jenkins, Time Series Analysis: Forecasting and Control, San Francisco: Holden-Day, 1970.
6) M. Khashei, M. Bijari, “A Novel Hybridization of Artificial Neural Networks and ARIMA Models for Time Series forecasting,” in Applied Soft Computing 11(2): 2664-2675, 2011.
7) A. Earnest, M. I. Chen, D. Ng, L. Y. Sin, “Using Autoregressive Integrated Moving Average (ARIMA) Models to Predict and Monitor the Number of Beds Occupied During a SARS Outbreak in a Tertiary Hospital in Singapore,” in BMC Health Service Research, 5(36), 2005.
8) Sak, Hasim, Andrew W. Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." (2014).