Predict Stock Prices with LSTM

Sherzod Muminov
Department of Computer Science
University of Illinois, Chicago
smumin2@uic.edu

Nihal Pratap Ghanathe
Department of Computer Science
University of Illinois, Chicago
nghana2@uic.edu

Bharath Koneti
Department of Computer Science
University of Illinois, Chicago
Bkonet2@uic.edu

Project Link: https://github.com/BharathKoneti/AIMACS594

1. Abstract

There are a lot of complicated financial indicators and the fluctuation of the stock market is highly volatile. However, as the technology is getting advanced, the opportunity to gain a steady fortune from the stock market is increased and it also helps experts to find out the most informative indicators to make a better prediction. The prediction of the market value is of great importance to help in maximizing the profit of stock trade while keeping the risk low.

Long Short-Term memory is one of the most successful RNNs architectures. LSTM introduces the memory cell, a unit of computation that replaces traditional artificial neurons in the hidden layer of the network. With these memory cells, networks are able to effectively associate memories and input remote in time, hence suit to grasp the structure of data dynamically over time with high prediction capacity.

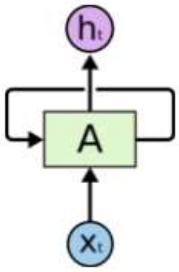
We are trying to predict stock closing price based on LSTMs. We collected a sample and used it for training and validation purposes for the model.

Keywords: Long short-term memory (LSTM), Gated Recurrent Unit (GRU), recurrent neural network (RNN), NYSE, root mean square error (RMSE), prediction and stock prices.

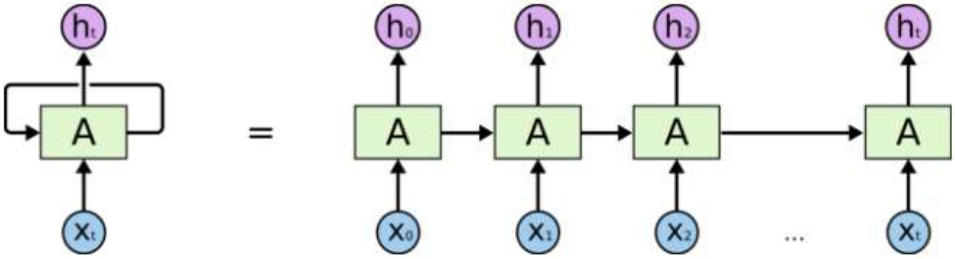
2. Methodology

Various types of neural networks can be developed by the combination of variegated factors such as network topology, training method etc. In this experiment we are using Long Short-Term Memory. We are also going to compare LSTM to other popular methodology – GRU (Gated Recurrent Unit).

LSTM is a neural network which can be used to predict and process at any point, which the traditional neural network cannot handle. They are networks with loops:

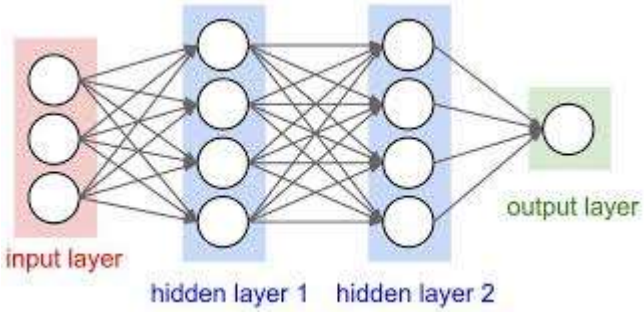


In this diagram, neural network A takes as input ‘xt’ and produces output ‘ht’. Loop makes it possible to pass information from one state of the network to the next step. Next steps are actually similar to their predecessors, and you can assume that you receive input ‘xt’ and producing output ‘ht’ at every step:



Above diagram represents unrolled version of a particular neural network A.

Neural Network architecture:



The architecture of our solution looks like as show in the above figure. There is an input layer, two hidden layers each with 50 and 100 LSTM cells respectively and an output layer. The hyperparameters here are the number of hidden units, the drop-out rate, number of epochs, batch size. We found that for the below hyperparameters, we got the best results.

Drop-out rate – 20%
Batch-size- 100
Number of Epochs = 50
Number of hidden units in layer1 = 50
Number of hidden units in layer2 = 100

The code snippet of the model being initialized is as shown in the below image with the above parameters. Here we use a **Linear Activation function**, a **mean squared error** loss function and a **gradient descent optimizer**.

```
model = Sequential()

model.add(LSTM(
    input_dim=1,
    output_dim=50,
    return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(
    100,
    return_sequences=False))
model.add(Dropout(0.2))

model.add(Dense(
    output_dim=1))
model.add(Activation('linear'))

start = time.time()
model.compile(loss='mse', optimizer='rmsprop')
print ('compilation time : ', time.time() - start)
```

3. Data

Stage 1: Raw Data:

In this stage, the dataset consisting of S&P 500 companies is used for prediction of future stock prices

Ex1: The date from S&P500 index is shown below:

date	symbol	open	close	low	high	volume
1/5/2016 0:00	WLTW	123.43	125.84	122.31	126.25	2163600
1/6/2016 0:00	WLTW	125.24	119.98	119.94	125.54	2386400
1/7/2016 0:00	WLTW	116.38	114.95	114.93	119.74	2489500
1/8/2016 0:00	WLTW	115.48	116.62	113.5	117.44	2006300
1/11/2016 0:00	WLTW	117.01	114.97	114.09	117.33	1408600
1/12/2016 0:00	WLTW	115.51	115.55	114.5	116.06	1098000
1/13/2016 0:00	WLTW	116.46	112.85	112.59	117.07	949600
1/14/2016 0:00	WLTW	113.51	114.38	110.05	115.03	785300
1/15/2016 0:00	WLTW	113.33	112.53	111.92	114.88	1093700
1/19/2016 0:00	WLTW	113.66	110.38	109.87	115.87	1523500
1/20/2016 0:00	WLTW	109.06	109.3	108.32	111.6	1653900
1/21/2016 0:00	WLTW	109.73	110	108.32	110.58	944300
1/22/2016 0:00	WLTW	111.88	111.95	110.19	112.95	744900
1/25/2016 0:00	WLTW	111.32	110.12	110	114.63	703800
1/26/2016 0:00	WLTW	110.42	111	107.3	111.4	563100
1/27/2016 0:00	WLTW	110.77	110.71	109.02	112.57	896100
1/28/2016 0:00	WLTW	110.9	112.58	109.9	112.97	680400
1/29/2016 0:00	WLTW	113.35	114.47	111.67	114.59	749900

Ex2: The data for stock twits is shown below:

	message	sentiment
0	$SPY crazy day so far!	bearish
1	$SPY Will make a new ATH this week. Watch it!	bullish
2	$SPY $DJIA white elephant in room is $AAPL. Up 14% since election. Strong headwinds w/Trump trac	bearish
3	$SPY blocks above. We break above them We should push to double top	bullish
4	$SPY Nothing happening in the market today, guess I'll go to the store and spend some $.	bearish
5	$SPY What an easy call. Good jobs report: good economy, markets go up. Bad jobs report: no more	bullish
6	$SPY BS market.	bullish
7	$SPY this rally all the cheerleaders were screaming about this morning is pretty weak. I keep adding 2	bearish
8	$SPY Dollar ripping higher!	bearish
9	$SPY no reason to go down !	bullish
10	$SPY There are only a handful of stocks out there able to withstand what it about to occur.	bearish
11	$SPY looking for new highs after that small pullback..	bullish
12	$SPY come on in everyone, the water is warm	bullish
13	$SPY Waiting patiently for a reentry, but seems evasive today.	bearish
14	$SPY $DJIA $DIA https://www.youtube.com/watch?v=_qBdijK2iIc	bearish
15	$SPY No DOW 20k for you!	bearish
16	ðŸš¨ $SPY ðŸš¨	bullish
17	$SPY just keep buying there are no scenarios where this ever goes back down 228 by week end ..350	bullish
18	$ES_F $SPY â€ª#â€ŽTradingâ€¬â€¬ â€ª#â€ŽFuturesâ€¬ â€ªToday's trading report was bias-1 bullish.	bullish
19	$SPY Bulls know this means less rate hikes. Just wait for it. They will swoop in and buy the entire dip	bullish
20	$SPY BTD BTD!!!! What a gift of a dip.	bullish
21	$SPY Will not stop until it has reached new ATH	bullish
22	$SPY spot vix lowest close in 30 months. CUTE	bearish
23	$QQQ $SPY $TBT $TLT Top bond/stock market pundit Jeff Gundlach-bonds have peaked & stocks co	bearish
24	$TLT $SPY Yield curve flattening. 10 yr US Tsy down 20 BP Since "experts" talked about 3+% Rates bs	bullish

Stage 2: Data Preprocessing:

The pre-processing stage involves

a) Data discretization: Part of the data reduction but with importance, especially for numerical data.
b) Data transformation: Normalization
c) Data cleaning: Fill in missing values.
d) Data integration: Integration of data files.

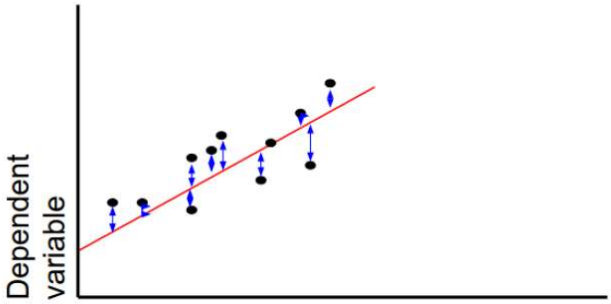
Stage 3: Feature Extraction

In this layer, only the features which are to be fed to the neural network are chosen. We will choose the feature from Date, open, high, low, close and volume.

4. Analysis

For analyzing the efficiency of the system, we have made use of the Root Mean Square Error(RMSE). The error or the difference between the target and the obtained output value is minimized by using RMSE value. The RMSE is the square root of the mean/average of the square of all the error. The use of RMSE is highly common and it makes an excellent general-purpose error metric for numerical predictions. RMSE amplifies and severely punishes large error.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum\left(\hat{Y}_i - Y_i\right)^2}$$



5. Experiment

For our project we have followed these steps:

1. Downloaded daily closing prices for stocks both form NYSE and NASDAQ for the period between 1/1/2000 and 12/31/2016.
2. We have implemented LSTM and GRU neural networks in order to predict stock prices
3. We have also used sentiment analysis of stocks based on TwitStocks
4. Other supporting functions in Python like plotting results were also implemented
5. Watson Conversation Services were used in order to interact with users with the help of Python SDK
6. All interaction with code and users is held in Jupyter Notebook

Data includes following fields:

a. Date
b. Ticker (name of the stock)
c. Closing price

We have preprocessed the data based on user input. For example, if user wants to make prediction for MSFT (Microsoft) stock, we have parsed data to include only MSFT results. Neural Networks are sensitive to the input data. So, we had to rescale data to achieve reasonable output.

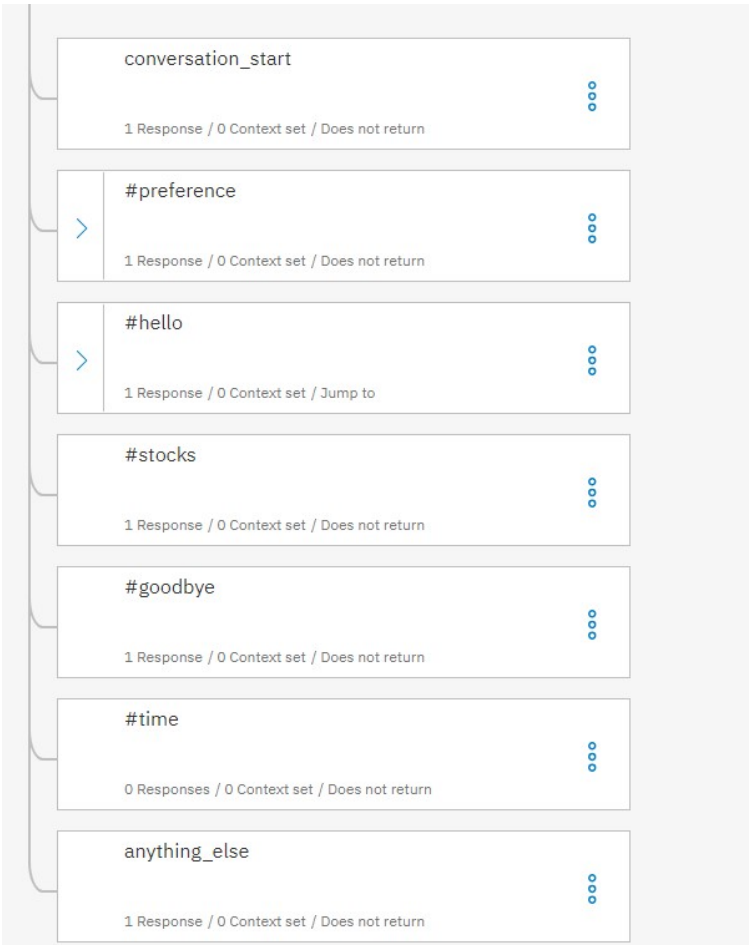
Watson

For the purpose of this project we have used Watson conversation services to interact with users. This makes usage of our project smoother and more enjoyable. Moreover, we have incorporated Watson API into our Jupyter solution, so that everything can be done directly from there.

Structure. We have created several intents for our conversation services:

- #hello –greets the user
- #methodused – recognizes method user wants to implement, this would be either LSTM or GRU
- #number – recognizes number of days user wants to predict stock prices
- #preference – recognizes what user wants to do. Currently, only stock prediction is implemented. But in the future other functionality may also be implemented
- #stocks – list of stocks that are supported by our service
- #goodbye –recognizes farewell messages, say bye to user and shuts down the program

Dialog consists of implementation of above intents



Further we have downloaded Python API for Watson Conversation Services and connected to our service directly from Jupyter:

```
import watson_developer_cloud

# Set up Assistant service.
service = watson_developer_cloud.AssistantV1(
  username = '9dcd5dac-d49b-42b0-85d6-4e5c0784ffd1', # replace with service username
  password = 'k4MH5tw21Qa1', # replace with service password
  version = '2018-02-16'
)
workspace_id = 'ef22da03-446b-45bc-a91d-2e39fb835dd9' # replace with workspace ID
```

We manipulated data in Jupyter, aka executed appropriated functions based on recognized intents by simple “if” statements. For example:

```
if(response['intents']==[{'confidence': 1, 'intent': 'methodused'}]):
  if(user_input=="LSTM"):
      methodu="LSTM"
  else:
      methodu="GRU"
```

In above code chunk, based on user input, we recognize #methodused intent and assign user_input into methodu variable, which is used as an argument in functions that has to be executed.

If we recognize #goodbye intent, we shut down all connections to Watson Conversation Services.

Output

Let’s review demo. When we run last code group in Jupyter we come across into dialog:

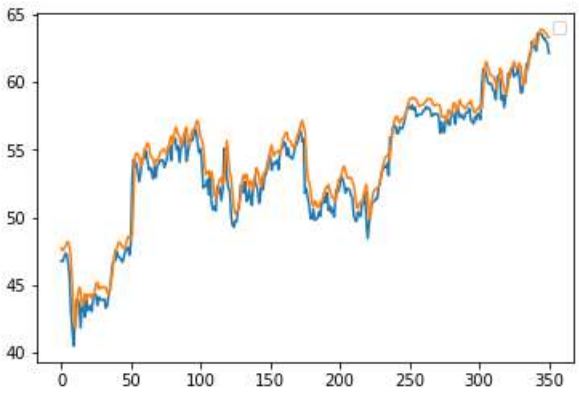
```
Hi. Welcome to stock value predictor. What would you like to do?
>> Predict Stock
How many days you want to see the forecast?
>> 5
Which method would you like to use (LSTM, GRU)?
>> LSTM
Which stocks forecast would you like to see? Please enter ticker number
>> MSFT
```

To first question we should specify our intent to predict stock value. Next, service asks us for how many days we want to make a prediction. Next, we choose method to use: either LSTM or GRU. And finally we specify stock ticker to predict.

Then, model trains and for each epoch it shows us loss value, which decreases over iterations:

```
1336/1336 [==============================] - 1s 857us/step - loss: 0.0546 - val_loss: 0.1535
Epoch 2/100
1336/1336 [==============================] - 0s 55us/step - loss: 0.0211 - val_loss: 0.0593
Epoch 3/100
1336/1336 [==============================] - 0s 54us/step - loss: 0.0119 - val_loss: 0.0270
Epoch 4/100
1336/1336 [==============================] - 0s 55us/step - loss: 0.0071 - val_loss: 0.0149
Epoch 5/100
1336/1336 [==============================] - 0s 53us/step - loss: 0.0028 - val_loss: 0.0046
Epoch 6/100
1336/1336 [==============================] - 0s 61us/step - loss: 8.6510e-04 - val_loss: 4.2244e-04
Epoch 7/100
1336/1336 [==============================] - 0s 52us/step - loss: 5.4045e-04 - val_loss: 0.0019
Epoch 8/100
1336/1336 [==============================] - 0s 50us/step - loss: 5.7778e-04 - val_loss: 3.6429e-04
Epoch 9/100
1336/1336 [==============================] - 0s 53us/step - loss: 5.9448e-04 - val_loss: 4.7986e-04
Epoch 10/100
1336/1336 [==============================] - 0s 56us/step - loss: 5.5837e-04 - val_loss: 0.0017
Epoch 11/100
1336/1336 [==============================] - 0s 52us/step - loss: 6.4273e-04 - val_loss: 0.0010
Epoch 12/100
1336/1336 [==============================] - 0s 58us/step - loss: 5.3509e-04 - val_loss: 0.0025
Epoch 13/100
1336/1336 [==============================] - 0s 55us/step - loss: 5.6956e-04 - val_loss: 0.0025
Epoch 14/100
1336/1336 [==============================] - 0s 57us/step - loss: 5.0097e-04 - val_loss: 2.4910e-04
Epoch 15/100
```

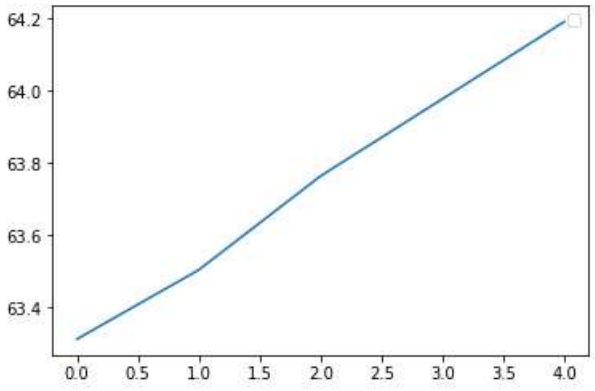
Finally, we receive an output with test (blue) and predicted (orange) values.



We also receive predictions for each particular day in the future. As well as error value, based on parameters entered previously:

```
The prediction for day 1 is 64.19177
The prediction for day 2 is 63.3128
The prediction for day 3 is 63.5045
The prediction for day 4 is 63.76444
The prediction for day 5 is 63.978073
Train Score: 0.81 RMSE
Test Score: 1.14 RMSE
```

Future price plot is also showed



At the end, if we type bye, program quits.

6. Code Flow:

First we have the **initialize** method that takes in the name of company whose stocks are to be predicted, number of day ahead of which the stock needs to be predicted and a parameter called “look_back”, which basically is the number of previous stock prices to be considered to predict the next stock price.

```
# split the data into test and train

    dataX, dataY = [], []
    for i in range(len(train)-look_back):
        a = train[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(train[i + look_back, 0])


    trainX = np.array(dataX)
    trainY = np.array(dataY)

    dataX, dataY = [], []
    for i in range(len(test)-look_back):
        a = test[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(test[i + look_back, 0])

    testX = np.array(dataX)
    testY = np.array(dataY)

    trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
    testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

Next based on the look_back parameter the data is divided into features and their true labels. Later we split the data into 80% training and 20% test data. And the model is initialized as explained above. Here

we have a choice of model to be used, i.e. LSTM or GRU. After the data is split into train and test, the model is trained with the above listed parameters.

```
model.fit(
trainX,
trainY,
batch_size=100,
nb_epoch=50,
validation_split=0.05)
```

Once the model is learnt, we use the function **predict_n** to predict the stock price 'Number' days ahead. i.e. previous 'look_back' stock prices are used to predict the stock of day 1 and next we use the look_back as well as the day 1 predicted value to predict the value for day 2 and so on till day n and a graph is plotted to show the stock values for the N predicted days.

```
# predicts stock price for nth day
def predict_n(number, testX, model,look_back):
    predicted = []
    arr = np.reshape(testX[len(testX)-1], (testX[len(testX)-1].shape[0], 1, testX[len(testX)-1].shape[1]))
    for i in range(int(number)):
        curr = model.predict(arr)
        for j in range(look_back-1):
            arr[0][0][j] = arr[0][0][j+1]
        arr[0][0][look_back-1] = curr
        predicted.append(curr)
    return predicted
```

Along with this predicted data, we also calculate the crowd sentiment of that company using StockTwits data, Where bearish meaning the stock sentiment is negative and bullish meaning the stock sentiment is positive. Here we calculate the number of bearish and bullish and based on the majority, the sentiment for that particular company is displayed. The function used to do that is as shown below.

```
def crowd_sentiment(company):
    stocktwits_data =  pd.read_csv('StockTwits_SPY.csv', header=0)
    senti = stocktwits_data.sentiment.values.astype('str')
    pos = 0
    neg = 0
    for i in senti:
        if i == 'bearish':
            neg += 1
        elif i == 'bullish':
            pos += 1

        if pos > neg:
            return "BULLISH"
        elif pos < neg:
            return "BEARISH"
        else:
            return "NEUTRAL"
```

7. Conclusion:

The popularity of stock market trading is growing rapidly, which encourages researchers and stock buyers to find alternative and better methods for the prediction using new techniques. The forecasting model requires accurate prediction for the models. The work presented has one of precise calculations using Long Short Term Memory unit which helps investors or analysts to better judge buying the stock.

References

[1] RakhiMahant, TrilokNathPandey, Alok Kumar Jagadev, and SatchidanandaDehuri —Optimized Radial Basis Functional Neural Network for Stock Index Prediction,‖ International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) – 2016.

[2] Kai Chen, Yi Zhou and FangyanDai —A LSTM-based method for stock returns prediction: A case study of China stock market,‖ IEEE International Conference on Big Data,2015.

[3] DhirajMundada, GauravChhaparwal, SachinChaudhari, and TruptiBhamare— Stock Value Prediction System,‖ International Journal on Recent and Innovation Trends in Computing and Communication, April 2015

[4] Hochreiter, Sepp, and Jürgen Schmidhuber— Long short-term memory,‖ Neural computation 9.8 (1997): 1735-1780