Stock Price Prediction Based on Past Price and Sentiment Analysis

Hao Zeng hz2759, Yuqing Jin yj2679, Shuai Ren, sr3849

Electrical Engineering
Columbia University
New York, USA
Email: {hz2759, yj2679, sr3849}@columbia.edu

1. Problem Statement

We implemented a method to predict the closing stock price by each day using Long Short-Term Memory (LSTM) and linear regression models based on tweet sentiment analysis scores.Our architecture is presented below:

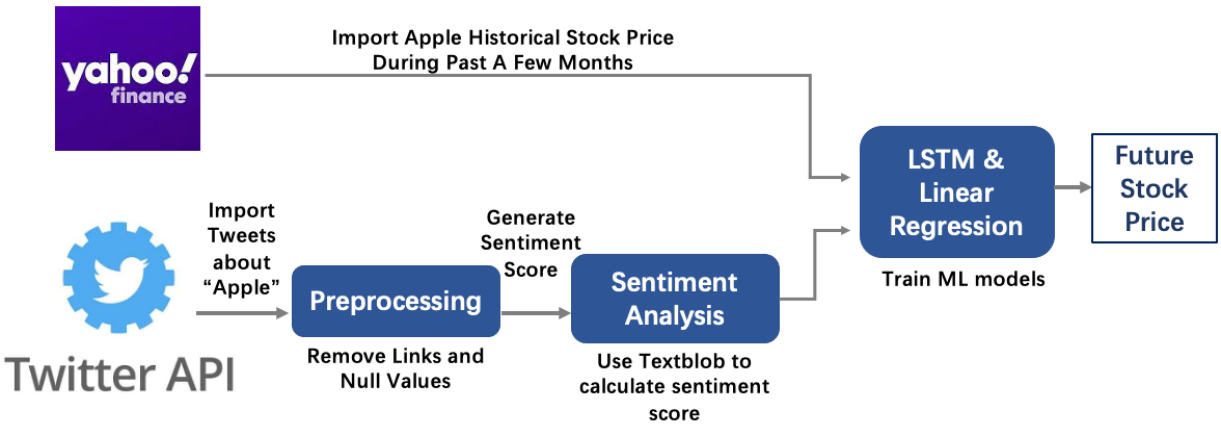

Figure1: The architecture of the project

(i) Stock data retrieval. By-day stock price scrapped from Yahoo Stock API. Extracted six main indicators about stock price as date, open, high, low, close, adj close and volume and period of past 3/6/12/24 months.

(ii) Twitter Sentiment Analysis. All tweets were scrapped from Twitter API. We kept the created time and tweet content according to the keywords, like "Apple", "iPhone" and "iPad". Data preprocessing and cleaning is to remove all the unnecessary content from the raw tweets. After that, we did structured streaming to combine the historical stock price and sentimental score.

(iii) Using two machine learning methods, LSTM and linear regression, to analyze the stream of the stock price and sentiment score.

Performance is based on Root Mean Squared Error (RMSE) value as well as stock price trend figure plotted with real price and prediction price.

2. Challenges Faced With

We came across challenges in Twitter API Crawling process, LSTM model training process and structure streaming process.

2.1 Twitter API Crawling Result

The first challenge we came across was that the maximum number of tweets for one retrieval is 500.

To solve this problem, we used next_token, count, and flag variables in a loop to retrieve more than 500 tweets in the same time period.

The second challenge in tweet crawling was that it is likely to retrieve all tweets at only the end of a period due to the number of tweets being huge for only one single day , e.g., if we want to retrieve 500 tweets from 10/01-10/31, the Twitter API would return 500 tweets created on 10/31

To solve this problem, We write a loop outside the retrieval process and used a list to store the time period to make sure that we could get tweets from the different time periods.

2.2 LSTM Model Performance

The LSTM performance on prediction was poor at first.

Therefore, we firstly tried a lag. Since it is possible that the sentiment score can have a layback effect on the close stock price, we tried to lag the close price by one, two, and three days respectively to see if the performance was better. The lag method did improve the performance a little but the performance of the model was still not satisfactory.

Our second attempt was to adjust the parameters, i.e., the number of RNN cells in the model as well as epochs and batch size. We finally have 10 RNN cells, 20 batch size, and 30 epochs (except for the 3-moths dataset, we had 50 epochs to train the model due to the limit of data). It also turned out that the lag of day did not have a significant impact on the performance compared to the adjusted model, so we canceled the lag technique.

2.3 Streaming Process

When we deployed the structured streaming part, we faced some difficulties. At first, we tried to implement all the sentiment analysis process in streaming, however, it did not work on both the local environment and Google Colab. We tried to adjust the output format to csv, console and memory, which did not work as well. So we decided to remove some parts from streaming. The data cleaning and score computation was done without streaming, they were executed and saved in csv files. Then we input the csv files into stream, and it works well to do the basic operations: groupby, sum up and sort. Our assumption is that former processes may contain some operations

not allowed by dataframe format or the csv files are inappropriate in tweet analysis streaming. These two assumptions may lead to the unacceptable process of overall sentiment analysis. In the future, we may explore how to run the whole process in streaming.

3. Technical Detail

The technical details we implemented are illustrated below.

3.1 Sentiment Analysis

For the sentiment analysis, there are three steps to implement it:

(i) Data Preprocessing & Cleaning

The goal of this step is to clean the unnecessary content of the original tweet content in order to make it easier to understand by computer. With the help of the Regular Expression module, we removed all the retweets, links, hashtags, video and audios from the raw tweet. Then we set all letters to lowercase and removed punctuations, double spaces and numbers. Finally, we lemmatized and tokenized content to simplify the content.

(ii) Sentiment Analysis Score Computation

Secondly, it is sentiment analysis calculation. We've used the textblob module to compute the polarity score of people towards Apple. Polarity is between -1 and 1: -1 represents negative feeling and 1 is positive feeling to Apple.

(iii) Combine sentiment score with historical stock price via Structured Streaming

At last, we used structured streaming to process the cleaned sentiment analysis result, in order to group by and sum up scores created on the same day and then combine it with the historical stock price according to their date.

3.2 LSTM

We used the tensorflow keras package to implement the LSTM structure.

Firstly, we input the raw dataframe, which has all the attributes of stock price, like "open", "high", "low", "volume", "close" and "sentiment_score". We set 80% data for training and 20% for testing. The input attributes are "open", "high", "low", "volume", and "sentiment_score" and the output attribute is "close". Finally, we input those attributes into MinMaxScalar to scale the data between 0 and 1; then the scaled data was sent to the LSTM model.

The architecture of LSTM model consists of three stages: the first one is 10 RNN cells, the second is the dense layer and the third one is the output layer.

A screenshot of the code implementing this process is shown as follows:

```
from keras import models, layers
model = models.Sequential()
model.add(layers.LSTM(10, input_shape=(1,5)))
model.add(layers.Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
print(xtrain.shape)
xtrain = xtrain.reshape((xtrain.shape[0], 1, xtrain.shape[1]))
xtest = xtest.reshape((xtest.shape[0], 1, xtest.shape[1]))
print('The shape of xtrain is {}: '.format(xtrain.shape))
print('The shape of xtest is {}: '.format(xtest.shape))

loss = model.fit(xtrain, ytrain, batch_size=20, epochs=30)
```

Figure2. The code for LSTM model

3.3 Linear Regression

We firstly created a dataframe called “StocksInfo" and selected 4 columns: “Open”, “Volume”, “Sentiment Score” and “Close”. Then we used two transformers: VectorAssembler called “assembler” and MinMaxScaler called “scaler” to form a pipeline. After fit() we could have a scaled model. Then we applied the transform() function to have a new dataframe with an assembled feature vector and data scaled to 0-1. The new dataframe was then fed into the linear regression model. The code for this part is attached below for better illustration:

```
assembler = VectorAssembler().setInputCols(['Open','Volume','Sentiment_Score']).setOutputCol('features')
scaler = MinMaxScaler(inputCol="features", outputCol="features_scaled")

pipeline = Pipeline(stages=[assembler, scaler])

lr = LinearRegression()
```

```
[ ] scalerModel = pipeline.fit(Data)
    scaledData = scalerModel.transform(Data)

    dataset=scaledData.select("Close",scaledData.features_scaled.alias('features'))
    dataset=dataset.withColumnRenamed("Close","label")
    dataset.show()


    train, test = dataset.randomSplit([0.8,0.2])
```

Figure3. The code for the linear regression model

4. Structured Streaming

This part demonstrates the streaming process we used and streaming algorithm for future study.

4.1 streaming process

In the process of structured streaming, each time it retrieves a certain amount of data to process the same operations. After processing, the data is appended to the continuously flowing data stream.

For our experiment, Spark firstly read the stream, which is the sentiment score table, there are 5 csv files we crawled from twitter in the past 3 months; after it read the stream, it was converted to dataframe type, then we specified the schema of our csv table. Then we wrote queries into the stream, which are groupBy, sum scores and sort by time. Next, we started streaming. During streaming, each time it processed one csv file at a time. At last, our results were saved in csv files as we set them up.

4.2 Further Study

For future work, we would like to (i) implement the streaming during the overall sentiment analysis processes and (ii) explore the parallel streaming to do real-time twitter sentiment analysis work to compare the future stock price prediction result of all the companies in the stock market.

To implement the real-time Twitter sentiment analysis, we can create a TCP socket between Twitter's API and Spark, which waits for the call of the Structured Streaming and then sends the tweets. After sentiment analysis, we can store the sentiment analysis scores in a parquet file, which is a data storage format. This process is referenced from the website, and it has a different input and output format from ours, which works better with real-time twitter streaming.

5. Future Work

In the future, we would like to make the following improvements.
(i) Stream the data ETL process, which means that we could build a pipeline with stages to retrieve data, calculate sentiment scores, and feed data to the machine learning model.
(ii) Implement real-time stock price prediction. We can retrieve recent tweets, make predictions with our machine learning model and make predictions of all the companies in the stock market in real-time. The goal is to predict the future trend of the stock price, which is practical in real life.