CS410 Final Project Documentation

(yning4 fteng3 zzhu24)

The final project we chose is to predict bitcoin's price given text data in reddit. The process is to firstly scrawl the data from reddit bitcoin's section and bitcoin's price from Nov 2016 to May 2018. Then based on the text data scrawled from the webpage, we will do sentiment analysis on it. With the score generated by sentiment analysis, we will then pass it into LSTM Model to learn the pattern between sentiment analysis score and bitcoin's price. Finally, the model will be able to predict future bitcoin's price.

1.Scrawl Reddit bitcoin's section text data and bitcoin price data

We used Quadl API to grab the bitcoin's price data. It requires a registered API key as shown. It also provides filters to select the data that the user wants. In this case, I filtered the data based on start date and end date. Then based on the data received, we then extract the data we need and format it for future use.

```
def get_bitcoin_price():
    data = quandl.get("BCHARTS/BITSTAMPUSD", authtoken="u4bnPE8zeaFZxMBNne6V", start_date="2016-11-01", end_date="2018-05-01")
    #print (data[['Open', 'Close', 'Weighted Price']])
    bitcoin_price_data = {}
    date_iter=datetime.date(2018,5,1)
    for i in range(len(data['Weighted Price'])):
        str_date = str(date_iter).replace("-", "")
        bitcoin_price_data[str_date] = data['Weighted Price'][str(date_iter)]
        date_iter -= datetime.timedelta(days=1)
    return bitcoin_price_data
```

For text data from reddit bitcoin's section, we firstly used a API from archive.org to retrieved the url from the specific date we pass in the the API call. Then with the url of the reddit bitcoin's section page of that date, we used an API called haven on demand to perform sentiment analysis. It will process the whole page and extract text data from that page. With the text data extracted, it will then classify the each sentences on the page towards either positive or negative with a trained model. Then with the result from classification, it will return a score indicating how positive or negative the sentiment towards bitcoin is on that date. We then format the data with the relative date to prepare for future merge with

bitcoin's price data.

```
def get_sentiment():
    date = datetime.date(2018, 5, 1)
    sentiment_score = {}
    while (date.year!=2016 or date.month!=10 or date.day!=31):
        month_string = str(date.month)
        if len(month_string) < 2:
            month_string = "0" + month_string
        day_string = str(date.day)
        if len(day_string) < 2:
            day_string = "0" + day_string
        date_string = str(date.year) + month_string + day_string
        r1 = requests.get("http://archive.org/wayback/available?url=reddit.com/r/bitcoin&timestamp=" + date_string)
        if(r1.status_code == 200):
            data1 = json.loads(r1.text)
            archive_url = data1['archived_snapshots']['closest']['url']
        else:
            archive_url = None
            print("Error return code = "+str(r1.status_code))
        r2 = requests.get("https://api.havenondemand.com/1/api/sync/analyzesentiment/v2?apikey=07af18eb-e943-4dd0-9fdf-93b06614c921&url="
            + archive_url, verify = False, timeout = 100)
        if(r2.status_code == 200):
            data2 = json.loads(r2.text)
            #sentiment_score['date'] =
            sentiment_score[date_string] = data2['sentiment_analysis'][0]['aggregate']['score']
        else:
            print("Error return code = "+str(r2.status_code))
        date -= datetime.timedelta(days=1)
    return sentiment_score
```

In this case, we select the reddit bitcoin's section text data and bitcoin's price data from Nov 2016 to May 2018

2. Perform prediction with scrawled data

With the formatted data in merged_data.csv

	Date	Price	Sentiment
2	20180501	0	0.04706125439649397
3	20180430	9272.9826038987	0.04706125439649397
4	20180429	9354.9324810618	0.04706125439649397
5	20180428	9238.00247349	0.04706125439649397
6	20180427	9199.1228425732	0.04706125439649397
7	20180426	8881.3737242717	0.04706125439649397
8	20180425	9178.4997966055	0.04706125439649397
9	20180424	9320.9705391561	0.04706125439649397
10	20180423	8890.2978063102	0.2466271691634673
11	20180422	8889.8999267592	0.3555639251092227
12	20180421	8799.3921152734	0.0663355353284589
13	20180420	8485.1246523128	0.18671078400233954
14	20180419	8210.256308339	0.28116365620901607
15	20180418	8076.6874334605	0.4116119467699309
16	20180417	8026.3879013008	0.17381126402667132
17	20180416	8097.6517274877	0.17475176552429608

With the bitcoin price of that day and total sentiment score of all the reddit post on that day.
We will let the user to choice how much percentage of the data they want to set as training data. Then, use the percentage to cut our data in training data and testing data. Theoretically, the more percentage we set to training, the more accuracy we will get.

```
21      percent = input('Please enter the percentage you want to set as training data. \n'
22                      'The percentage should be within [5,95]: ')
23      percent = int(percent)
```

Our idea is to use two previous price and today's sentiment score as input to predict today's price. Therefore our input data has three values: [day before yesterday's price, yesterday's price, today's sentiment so far]. Our desired output is today's closing price.

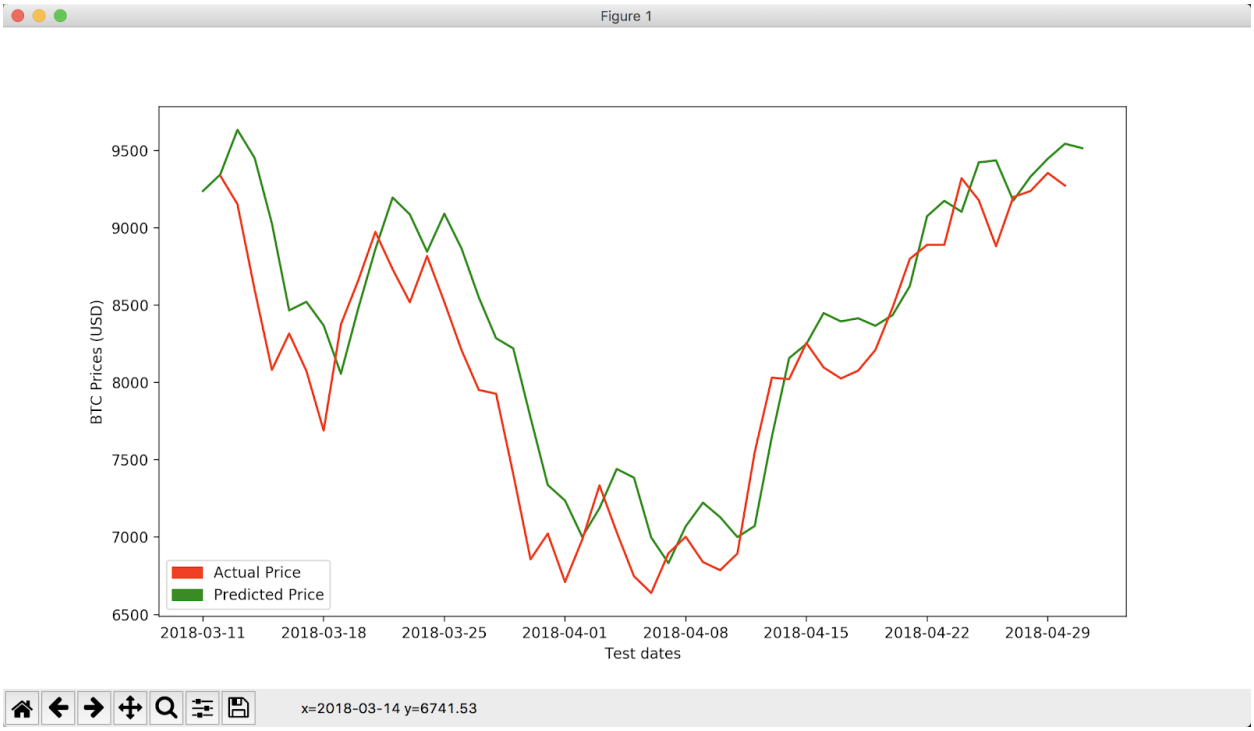
```
46      # process data to use [days] day prev [prev-price and this day's sentiment]
47      day = 2
48      Xtrain = []
49      Ytrain = []
50      for i in range(0, len(train_x)-day):
51          # price
52          y = train_y[i]
53          Ytrain.append(y)
54          # x = get 2 previous day's price
55          x = train_y[i+1:i+1+day]
56          # x = append today's sentiment score from reddit data
57          x = x.tolist()
58          x.append(train_x[i].tolist())
59          Xtrain.append(x)
60
61      train_x = np.array(Xtrain)
62      train_y = np.array(Ytrain)
63      print("Training input shape: ", train_x.shape)
```

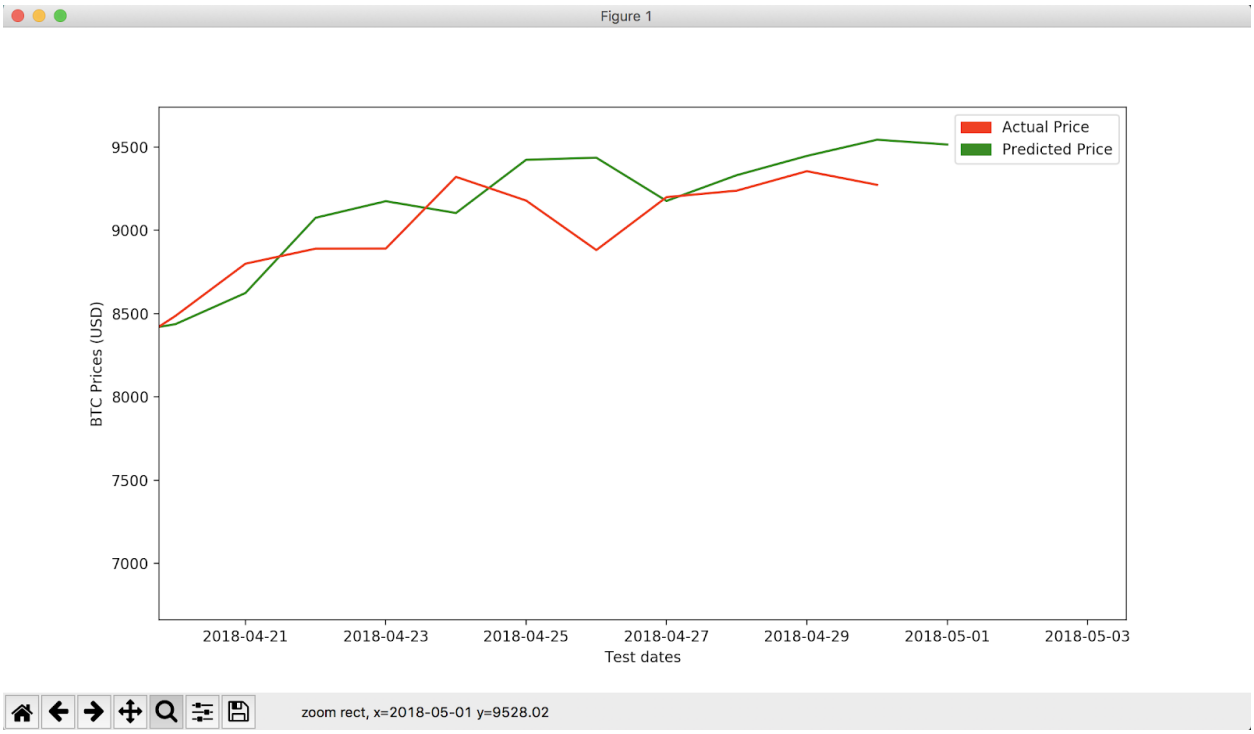
We selected LSTM Recurrent Neural Networks since we want to utilize its short term memory dependencies. We have tested other error functions and finally we choice to minimize Mean Absolute Error as our loss function because our final result is the price and its negative error or positive error does matter to the final result. In addition, 50 nodes are also the result of trial and error, which overall give us the highest accuracy. In addition, we take the advantage of Adaptive

Gradient Algorithm and Root Mean Square Propagation to optimize our LSTM Model. Finally, train the model with 300 epochs steps with batch size 100.

```
84      # Model building:
85      # LSTM Networks with
86      #   3 layers
87      #   50 nodes per layers
88      #   loss = Mean Absolute Error
89      #   optimizer = Adam Optimization
90      import keras
91      lstm = keras.models.Sequential()
92      lstm.add(keras.layers.LSTM(50, input_shape=(train_x.shape[1], train_x.shape[2]), return_sequences=True))
93      lstm.add(keras.layers.LSTM(50))
94      lstm.add(keras.layers.Dense(1))
95      lstm.compile(loss='mae', optimizer='adam')
96
97      # Model training:
98      #   bagging with
99      #   batch_size = 100
100     #   epochs = 300
101     re = lstm.fit(train_x, train_y, epochs=300, batch_size=100, validation_data=(test_x, test_y), verbose=0, shuffle=False)
```

When we set 85 as training data, we could be able to mean square error as 0.069 for raw data. Raw data means price after normalization with range = [0,1]. We have graph the test data's price output in comparison as the actual data as follow:





For example, our last time getting data was on may 1, 2018. We get the predicted price printed in terminal = “[May 01, 2018] predicted BTC price (USD) = [9515.156]”

3. Usage:

For this project, we use python 3.6 version. To run the train.py, we require python 3.6 or more advanced version.

API usage: Credit to

- Quandl API,
- HavenOnDemand API,
- web archive API

Python packages:

- Matplotlib
- datetime
- time
- sklearn.metrics
- keras
- sklearn.preprocessing

math
numpy
pandas
quandl
json
datetime
requests
csv
pprint
selenium

4. Teamwork:

Ideas: Tengfei, Yayi, Zhiyu
Get data: Tengfei, Zhiyu
Clean data: Tengfei, Zhiyu
Modeling: Yayi, Zhiyu
Debug: Tengfei, Yayi, Zhiyu
Project report: Tengfei, Zhiyu
Video: Tengfei, Yayi