INFS 772

Project 1

Due: 3/5/2015

This project can be done individually or in groups. Each group includes at most 2 people.

**Project overview:**

I have downloaded some tweets regarding the stock BAC (Bank of America) from StockTwits **(**[**http://stocktwits.com/**](http://stocktwits.com/)**)**, a twitter-like social media platform designed for sharing ideas between investors and traders. These tweets were posted between 2013-01-01 and 2013-02-26 and are stored in a json file (BAC.json). You need to conduct sentiment analysis of the tweets and show the results of the sentiment analysis using graphs.

**The objectives:**

1. Practice general-purpose Python programming
2. Get familiar with the python Datetime package
3. Learn the basics of data visualization using matplotlib

**Implementation details:**

You need to implement the following functions

1. **read\_stocktwits():**

In this function, you need do the following things:

1). Load the Json file to obtain a list of python dictionaries. Each dictionary represents a tweet and its metadata regarding the stock BAC.

2). Extract three things from each dictionary: the date/time of each tweet ("created\_at"->"$date"), the tweet content(“body”), and the user-assigned sentiment value of each tweet (["entities"]["sentiment"]["basic"]).

3) Process the tweet content. Hint: you want to remove the punctuations in each tweet. A method that is useful for doing this is char.isalnum(). You can convert the tweet content string into a list of character. You then call char.isalnum(). If a character is not alnum(i.e., an alphabetic or a number), you can remove it. You can then convert the list of characters back to a string. You also need to encode each tweet into the ASCII format and lower-case each word in each tweet.

4) The date/time field in the json file is a timestamp. Please find out how to convert a timestamp into a date/time string. You want to use python’s datetime package. A tutorial for the datetime is available at <http://pleac.sourceforge.net/pleac_python/datesandtimes.html>. Please read the tutorial first.

4) A user-assigned sentiment value can be “Bullish”, “Bearish” or “null”. Please replace “null” with “Unknown”.

5) For each dictionary (representing information related to a tweet), you need to create a list that includes the date/time, the processed tweet content, and the user-assigned sentiment. You then need to write the lists to a csv (comma separated) file.

6) This function does not have inputs and outputs. It will write the lists to a csv file (“BAC.csv”). Your csv file should be similar to “BAC.csv” in the assignment zip file.

1. **sentiment\_analysis():**

In this function, you need to do the following things.

1. Read data from BAC.csv created via read\_stocktwits():
2. If the user-assigned sentiment is “Bullish” or “Bearish”, you don’t need to conduct sentiment analysis for the tweet. You trust users’ inputs.
3. If the user-assigned sentiment is “Unknown”, you need to conduct sentiment analysis for the tweet. In the zip file there are two files "positive\_words.txt" and "negative\_words.txt", each of which contains a list of positive or negative words. Given a tweet, you need to count the number of positive words vs the number of negative words in the tweet. If the numbers are equal or they are both 0, you designate the sentiment of the tweet to be “Neutral”. If there are more positive words than negative ones, the sentiment is “Bullish”, and “Bearish” otherwise.
4. You then create another csv file (BAC2.csv). Each line includes two fields separated by a comma – the date/time of a tweet and its sentiment. You csv file should look similar to “BAC2.csv” in the assignment zip file.
5. Hint: When you write data to a csv file line by line, please remember to add a new line (\n) to each line. When you read data from a csv file line by line, please remember to remove the new line (You can call string.strip() to do so)
6. This function also does not have any inputs or outputs.
7. **get\_sentiment\_dates(start\_date, end\_date)**

In this function, you need to do the following things.

1. This function has two parameters, start\_date and end\_date. You need to read data from the BAC2.csv file first and then select the tweets that were posted between start\_date and end\_date (both are inclusive). For example, if start\_date is 2013-01-01 and end date is 2013-01-10, you need to select all tweets that were posted between 2013-01-01 00:00:00 (inclusive) and 2013-01-11 00:00:00 (exclusive).
2. You **MUST** convert start\_date, end\_date, and the post date/time of each tweet into python datetime objects and then select tweets based on python datetime comparisons. You are not allowed to do this using string manipulations (Actually, if the start time string is “2013-01-01” and end date is “2013-01-10”, and the post time of an tweet is “2013-01-06”, you can determine if the post time is between the start time and the end time by doing string comparisons. This, however, is not allowed in this project, because one of the objectives of the project is to get you familiar with the python date/time functions). Please read the datetime tutorial <http://pleac.sourceforge.net/pleac_python/datesandtimes.html>.
3. The output of the function is a list of dictionaries. There are three dictionaries in the list: positive\_dict, negative\_dict, and neutral\_dict. These three dictionaries include, respectively, the counts of positive, negative and neutral tweets on each day. As an example, the keys in positive\_dict represent the dates between start\_date and end\_date (both inclusive), and the values represent the counts of positive tweets on each day. If there are no positive tweets on a given day, the value should be 0. The negative\_dict and neutral\_dict are constructed similarly. Negative\_dict includes the counts of negative tweets on each day, and neutral\_dict the counts of neutral tweets on each day. A test case is provided in the code skeleton.
4. **drawing\_pie(start\_date, end\_date):**

In this function you need to do the following:

1. You need to call the get\_sentiment\_dates(start\_date, end\_date) function and obtain the three dictionaries.
2. If you call drawing\_pie('2013-01-02', '2013-01-31'), you should see a pie chart that looks similar to “pie\_sentiment.png” in the zip file. You don’t need to worry if the title of chart and the label “positive” overlap in your pie chart.
3. Note that the title of the chart indicates the “overall\_sentiment” within the period. The logic for computing this overall\_sentiment is simple: It is positive if there are more positive tweets than negative or neutral tweets, negative if more negative tweets than positive or neutral tweets, neutral if more neutral tweets than positive or negative ones. If there are equal numbers of positive vs. negative tweets, then the overall sentiment is neutral. If there are equal numbers of positive and neutral tweets and fewer negative ones, the overall sentiment is positive, and if equal numbers of negative and neutral tweets and fewer positive tweets, the overall sentiment is negative.
4. You need to use matplotlib to draw the pie chart. Please watch the video https://www.youtube.com/watch?v=P7SVi0YTIuE (The tutorial document is available at <https://scipy-lectures.github.io/intro/matplotlib/matplotlib.html>) to learn the basics of matplotlib. You can skip sections including 1.4.3.4 Ticks, 1.4.4.4. Contour Plots, 1.4.4.5. Imshow, 1.4.4.7. Quiver Plots, 1.4.4.8. Grids, 1.4.4.10. Polar Axis, and 1.4.4.11. 3D Plots. The author did the tutorial using the pylab package. In our project, you need to use matplotlib.pyplot. These two packages are very similar. Most methods in pylab are also available in pyplot.

Another short but useful tutorial is <http://matplotlib.org/users/pyplot_tutorial.html>.

1. Hint: When I draw a chart, I usually go to the matplotlib gallery <http://matplotlib.org/gallery.html> to find a graph that looks similar to what I want to draw and then copy and modify the code.

For example, an example of a pie chart can be found at <http://matplotlib.org/examples/pie_and_polar_charts/pie_demo_features.html>. I can then copy and modify the code. In the youtube tutorial, the author passed numpy arrays as input data to matplotlib. Actually, matplotlib can take any array-like data objects (e.g., list, tuples, pandas series) as inputs.

1. **drawing\_lines(start\_date, end\_date):**

In this function you need to do the following:

1. You need to call the get\_sentiment\_dates(start\_date, end\_date) function and obtain the three dictionaries.
2. If you call drawing\_lines('2013-01-02', '2013-01-31'), you should see a line chart that looks similar to “lines\_sentiment.png” in the zip file. The x-axis of the line chart represents the dates between start\_date and end\_date, and the y\_axis represents the counts of positive, negative, and neutral tweets.
3. By looking at the matplotlib gallary <http://matplotlib.org/gallery.html>, I found one diagram that looks close to the line graph I want to draw <http://matplotlib.org/examples/api/date_index_formatter.html>.

I don’t need to do something that is too fancy. I just want to modify the code below

r.sort()

r = r[-30:] *# get the last 30 days*

*# first we'll do it the default way, with gaps on weekends*

fig, ax = plt.subplots()

ax.plot(r.date, r.adj\_close, 'o-')

fig.autofmt\_xdate()

Here, r is a numpy record array (You don’t need to worry about what a record array is). A record array is similar to a dictionary, so in the code above ax.plot(r.date, r.adj\_close, ‘o-’), two arrays (date (like “keys” in a dictionary) and adj\_close (like “values” in a dictionary)) are extracted from the record array and the program draw a line chart with date on the x axis and adj\_close on the y axis. In our case, we can extract the keys (i.e.,dates) and the values (i.e., counts) from a dictionary and draw a line chart. Please pay attention that r.sort() is called in the code above. In our project, we can get the dates by simply calling dates = dict.keys(), but remember, we want to sort the dates, and then based on the sorted dates we get a list of counts (dict.values() will not work if the values are sorted based on dates),

1. Please watch the matplotlib tutorial to find out how to plot multiple lines on one line chart and how to add legend to the line charts.

Please work on the file project1.py. In the main method, I have created test cases for each function. If you see your results are slightly different from the results I gave you, don’t worry. It’s possible that we have used different methods for processing the contents of the tweets. You are not going to lose points, as long as you use reasonable methods for removing the punctuations (we can do this in different ways, which may generate slightly different results), do the encoding, and lower-case the words in each tweet. In addition to the above 5 functions, you can also develop any utility/helper functions deemed helpful.

Also, you need to install different packages including numpy, scipy, matplotlib, six, pyparsing etc. for this project. To make things simple, I put all the packages you need to use in the folder “packages”. Please open the folder and copy all the files/directory in the folder and paste them to C:\Python27\Lib\site-packages\

.