

Twitter Data Analysis - Milestone 3

Introduction

In this document I will briefly introduce the program, which aim on collecting analytic data from one of the largest social network called Twitter. To do so we will be using Twitter API, which is some kind of a wrapper for Python that is able to extract data from Twitter. The program prepare all needed tools and then takes care of the following questions:

1. Derive the sentiment of each tweet using Python module
2. Top 10 hashtags and users based on their number of tweets in your data set
3. Get the followers of a given twitter user from your acquired data set.
4. Given a twitter user, obtain the tweets and profiles of all followers of the user and show it.

Initializing

Part I

First of all, one need to create a twitter developer account and obtain the credentials. We need to visit this site¹ and fill up some forms in order to have access to creating new app in Twitter API. While creating an app I have used the name "OTH - Twitter Data Analysis". All of the tasks require working Twitter API for Python. I will work with the one called Tweepy.

User will execute commands in Jupyter Notebook. The first thing user need to do is run script **1.1 Credentials**, which puts keys and tokens into a dictionary and create a JSON file. The writing is made by lines below. I will let those keys and tokens filled up with my Twitter API credentials because it is a university project. This script and every next one is ended with printing "Done!" so the user knows once the computing is already finished.

```
with open("twitter_credentials.json", "w") as file:
    json.dump(credentials, file)
```

Part II

Then, user proceed to script **1.2 Initialize**. In the beginning, all needed imports need to be loaded into the virtual machine. It asks him about event, which he want to look up, using method `input()`. Hashtag sign is already in the prompt and given text is transferred to lower case by method `lower()`. Then user is supposed to write the oldest possible date of tweets in given format and maximum number of fetched tweets. For number of tweets there is possibility to omit the parameter if the user want all the results, but it can take a long time. User can decide whether he wants full statistic data or only small sample of data. Sometimes when there is too much requests per time there might appear following disclaimer below to inform user that the program need to wait a certain amount of time in seconds to continue requesting Twitter API. It is not a bug, but the limitation of Twitter API.

```
Rate limit reached. Sleeping for: #
```

¹ Twitter Developer Account: <https://developer.twitter.com/en/apps>

In next step, the Twitter API credentials are load into `auth` using similar method `json.load()`. Then we prepare tools for querying. As you can see below method `API` has several parameters which take care of not exceeding pagination limit of Twitter API. If the user is inactive for a long period of time Jupyter refreshed itself user need to execute this command again.

```
auth = tw.OAuthHandler(creds["CONSUMER_KEY"], creds["CONSUMER_SECRET"])
auth.set_access_token(creds["ACCESS_TOKEN"], creds["ACCESS_TOKEN_SECRET"])
api = tw.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

Then we need to load data set. It will be done with method `tweepy.Cursor` with given search formula that will look like this `#hashtag -filter:retweets`, where we get rid of retweets so data are not distorted. We will be executing this method only once because of Twitter rate limits of requests. You can see the whole method below.

```
tweets = tw.Cursor(api.search, q = SEARCH_EVENT, = "en",since =
                    SINCE_DATE).items(ITEM_NUM)
```

In the command we asked about searching the Twitter with our query I have mentioned earlier. Parameter `api.search` states that we are looking for tweets. We search only in English with given filtering (date and max count). Then we process our data set and save it to pandas DataFrames using several for cycles. Each column except the one of hashtags is fetched using cursor accessing function `tweet.object.attribute`. To get list of all hashtags I am using the method `tweet.entities.get("hashtags")`. We will be using `df_tweets` in next scripts as well.

```
tweets_dataset=[]
for tweet in tweets:
    hashtags = tweet.entities.get("hashtags")
    tags=[]
    for hashtag in hashtags:
        tag = hashtag['text'].lower()
        tags.append(tag)
    tweets_dataset.append([tweet.author.screen_name,tweet.author.name,
                           tweet.text,tags])
df_tweets = pd.DataFrame(tweets_dataset,
                          columns=["user","real_name","text","hashtags"])
```

Question 1 – Sentiment

The first question is about extracting sentiment (polarity) of each tweet in my dataset. User get the results by running script 2.1 `Derive the sentiment`. After we have loaded our data set to DataFrame we need to clean our data so we will have better final results of polarity. We have defined method below which takes String value of text of the post as a parameter and returns the text without non-alphanumerical chars. It uses `regex` to remove all URL addresses and all characters which are not numbers, letter or blank spaces (space or tabulator).

```
def remove_url(text):
    return " ".join(re.sub("(\w+:\/\/\w+)|([^\0-9A-Za-z\t ])", "",
```

```
text).split())
```

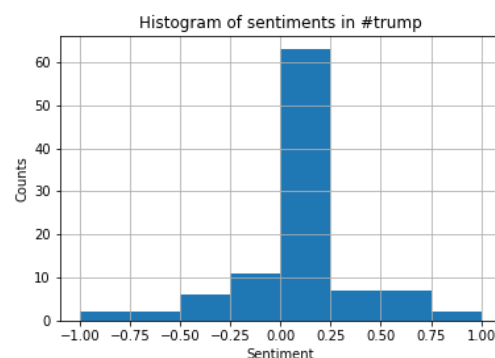
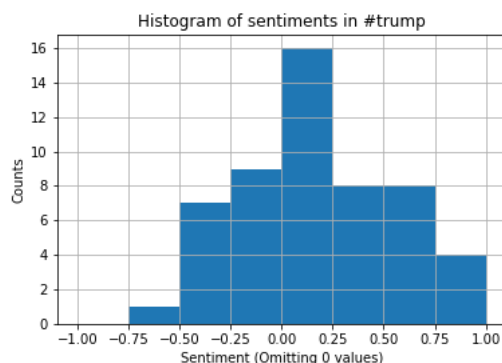
After that we will create an array of sentiment objects where each object relate to one text of tweet.

```
sentim_obj = []  
for tweet in clear_tweets:  
    sentim_obj.append(TextBlob(tweet))
```

Then we make the similar process but we create an array of pairs (text value of tweet and its sentiment value).

```
sentim_values = []  
for tweet in sentim_obj:  
    sentim_values.append([str(tweet),tweet.sentiment.polarity])
```

After that we create DataFrame from this array of pairs for easier access. Now we can display data visually using histogram. Because value of sentiment is somewhere on given scale $<-1,+1>$ we can create bins with reasonable values. We set title and axis, allow grid and we are ready to save the plot as PNG and show it to the user. In order to display histogram correctly without distortion I have decide to filter out all undecided sentiments whose value equal zero. This gives us better visual representation of actual polarity of tweets. I have provide a quick look how it would look like if I skipped this step as you can see on the right side.



Then we just simply save more detailed data of sentiments including zeros values using panda's method `to_csv` where we use semicolon as separator. To display results inside Jupyter we use method `display()` which is far better for showing DataFrames rather than `print()`. We also make sure the DataFrames is showing all the rows to the user not only head and tail when there is a lot of records by `pd.set_option("display.max_rows",size)`.

```
clear_sentiment_df = sentiment_df[sentiment_df.sentiment!=0]  
...  
ax.hist(x,bins=[-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1])  
...  
sentiment_df.to_csv("01_sentiment_res.csv",index=False,sep=";")  
display(sentiment_df)
```

Question 2 - Hashtags and users

Part I

In the second question we need to come up with Top 10 most used hashtags and Top 10 users based on number of tweets. All of it is related to our fetched dataset according to searched event. This is done by script 2.2 Top 10 hashtags and users. In the beginning I will start with Top 10 hashtags. The main algorithm of this script is supposed to count occurrence of each hashtag. For this purpose I have used a dictionary and iterating through all tweets using for loop. I iterate each hashtag of this list and deciding whether it is already in my dictionary or not. Based on that I either adding 1 if the tag is already in there or setting its value to 1 if not. In order not to distort the results I have transferred all the hashtags to lower case before proceeding.

```
hashtags_dict={}
for i in range(len(df_tweets)):
    hashtahgs = df_tweets.loc[i,"hashtags"]
    for tag in hashtahgs:
        if tag in hashtags_dict.keys():
            hashtags_dict[tag] += 1
        else:
            hashtags_dict[tag] = 1
```

Next step is come up with Top 10 hashtags with highest occurrence. That is done by following bunch of code.

```
sorted_users_dict = sorted(users_dict, key=users_dict.get, reverse=True)
for hashtag in sorted_hashtags_dict:
    if(i==10):
        break
    top_hashtags_dict[hashtag] = hashtags_dict[hashtag]
    i+=1
```

Firstly I sort the dictionary based on its occurrence value using method `sorted()` and then I take first 10 hashtags and put them to new dictionary. After that I simply creating an array with two pairs (hashtag and occurrence) and put it right into a DataFrame for nicer appearance. Right now I am able to display results with DataFrame and also save it to CSV file with method `to_csv`.

Part II

When it comes to second part of the question about fetching Top 10 users I will only make some points on the code which differs because a lot of algorithm and procedures are quite similar to looking for Top 10 hashtags.

Instead of only one dictionary I will be using two of them because I want to display not only the username and its occurrence but also their real (or stated) names. First dictionary will be the same as before for handling number of occurrence. The second one will map user as a key to its real name as a value. For accessing those values I will use already known `tweet.author.*` syntax from Twitter API.

```

for i in range(len(df_tweets)):
    user = df_tweets.loc[i,"user"]# @user
    name = df_tweets.loc[i,"real_name"]# alias
    if user in users_dict.keys():
        users_dict[user] += 1
    else:
        users_dict[user] = 1
        names_dict[user] = name

```

Next I will execute almost the same bunch of code where I only added double mapping because of real name. In the end I will again create DataFrame, save results to CSV and display them to the user.

All the results are displayed to user in Jupyter in DataFrames and also stored as CSV file to disk as it is done with all the tasks.

#hashtag	occurrence
0	biden 345
1	trump 58
2	usa 18
3	china 17
4	covid19 12
5	us 11
6	joebiden 11
7	america 10
8	bidenharris 9
9	johnson 8

Looking for Top 10 users...
CSV file of Top 10 users was created!

@user	real_name	occurrence
0	ChikaEgwurube	Chika Egwurube 21
1	SinaMoghtader	Sina Moghtader 6
2	jphoganorg	Hogan 5
3	breaknews1	Break'n News 4
4	Leonard5954150	Leonardo 4
5	moughthere	MissCrayon 4

A	B	C
1	#hashtag	occurrence
2	biden	345
3	trump	58
4	usa	18
5	china	17
6	covid19	12
7	us	11
8	joebiden	11
9	america	10
10	bidenharris	9
11	johnson	8
12		

Question 3 - Followers

Our main goal in this task 2.3 Followers of a given user was to get the followers of a given twitter user from the acquired data set. Firstly, we needed to access our data set using these lines.

```

for i in range(len(df_tweets)):
    users.append(df_tweets.loc[i,"user"])

```

After that we have showed user a table with all possible users in our data set and asked him about which one do he wishes to examine using method `input()`. While creating this table I have delete all duplicates using `pd.drop_duplicates()`.

Then I have used Tweepy's `Cursor()` function to get set of the followers while applying the given user as a parameter. Parameter `api.followers` looks for followers with limiting number of results by using method `.items()` in the end.

```

users = tw.Cursor(api.followers,
                  screen_name = USER_NAME).items(USER_COUNT)

```

Now I have proceed with extracting username and stated name.

```

for user in users:

```

```
followers.append([user.screen_name, user.name])
```

After that I have created DataFrame as in tasks before, print in using the method `display()` and save it to CSV file using the method `.to_csv()`.

```
43 print("Done!")
44
```

	@user	real_name
0	ChocolateDivk	Chocolate_divk
1	DetroitHole	The Detroit GloryHole
2	elizabeth_ruby4	Elizabeth
3	CullenNeed	Cullen_need_man
4	RayRay2456	Chocolate
5	Chicken1N	Pressure
6	freaky19901990	mosthated313

```
43 readerbythesea
44 PlatoCurse
45 WeWillBeBack8
46 DailyResister
47 Cor_de_Prensa
48 MichaelJamesFr8
49 Chrisk1100rs
50 MohdMuzzammilK
51 ReptilianSigh

Please provide a username to examine his followers: KayCee_313
Please provide a max number of results (Empty for no limit):
Looking for followers...
CSV file of followers of the user @KayCee_313 was created!
```

Question 4 – Tweets and profiles

Part I

In the last task 2.4 Tweets and profiles of all followers we were supposed to obtain the tweets and profiles of all followers of the given user and show it. After getting data set and showing the table of users as I have already showed in the previous task, we have ask about the user to fetch his tweets and profiles of his followers. We have used `Cursor()` method with parameter `api.user_timeline` that will look for timeline of given user. The number of results are again limited by `.items()` function.

```
tweets = tw.Cursor(api.user_timeline,
                    id = USER_NAME).items(TWEET_COUNT)
```

After that we simply transfer text messages of each object using `tweet.text`. Using for we are now able to put all our tweets of given user from `Cursor` to `Dataframe` using Python array and method `pd.DataFrame()` as shown below.

```
for tweet in tweets:
    tweets_arr.append(tweet.text)
df_texts = pd.DataFrame(tweets_arr, columns=["text"])
```

Now we only print results and save to CSV and we are done with the first part of this task

Part II

If we want to get profiles of followers the process is similar but we are accessing different values from the `Cursor()`. And also cursor has now parameter `api.followers`.

```
users = tw.Cursor(api.followers,
                  screen_name = USER_NAME).items(PROFILE_COUNT)
```

Now we use our algorithm to get user name, stated name and profile as a conjunction of the string `https://twitter.com/` and the found username `user.screen_name`.

```
for user in users:
    profile = f"https://twitter.com/{user.screen_name}"
    profiles_arr.append([user.screen_name, user.name, profile])
```

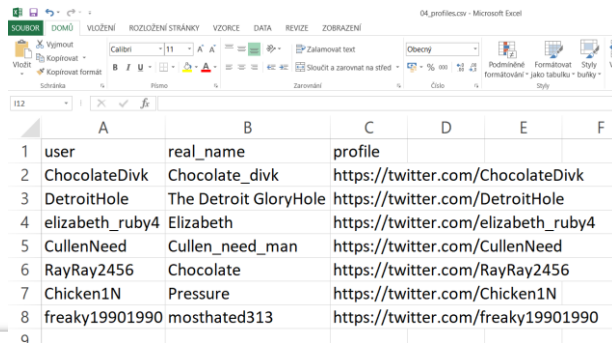
```
df_profiles = pd.DataFrame(profiles_arr,
columns=["user", "real_name", "profile"])
```

Our job is almost done we just need to print and save the results again. Output should look like this.

Done!
Looking for profiles of given user...
CSV file of profiles of the user @KayCee_313 was created!

	user	real_name	profile
0	ChocolateDivk	Chocolate_divk	https://twitter.com/ChocolateDivk
1	DetroitHole	The Detroit GloryHole	https://twitter.com/DetroitHole
2	elizabeth_ruby4	Elizabeth	https://twitter.com/elizabeth_ruby4
3	CullenNeed	Cullen_need_man	https://twitter.com/CullenNeed
4	RayRay2456	Chocolate	https://twitter.com/RayRay2456
5	Chicken1N	Pressure	https://twitter.com/Chicken1N
6	freaky19901990	mosthated313	https://twitter.com/freaky19901990

Done!



The screenshot shows a Microsoft Excel spreadsheet with the following data:

	A	B	C	D	E	F
1	user	real_name	profile			
2	ChocolateDivk	Chocolate_divk	https://twitter.com/ChocolateDivk			
3	DetroitHole	The Detroit GloryHole	https://twitter.com/DetroitHole			
4	elizabeth_ruby4	Elizabeth	https://twitter.com/elizabeth_ruby4			
5	CullenNeed	Cullen_need_man	https://twitter.com/CullenNeed			
6	RayRay2456	Chocolate	https://twitter.com/RayRay2456			
7	Chicken1N	Pressure	https://twitter.com/Chicken1N			
8	freaky19901990	mosthated313	https://twitter.com/freaky19901990			