Twitter Data Analysis - Milestone 2

In this write up I will briefly introduce part of the program, which prepare all needed tools and then takes care of first two 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

Preparing tools

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

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

Then user proceed to script `1.2 Initialize`. Firstly, 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()`. Loading dataset in next steps 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. Then user is supposed to write the oldest possible date of tweets in given format and maximum number of fetched tweets. User can decide whether he wants full statistic data or only small sample of data. In next step the Twitter API credentials are load to variable `auth` using similar method `json.load()`. Then we prepare tools for querying as you can see here where `wait_on_rate_limit=True` is taking care of not exceeding pagination limit of Twitter API. If the user is inactive for a long time and Jupyter refreshed itself user need to repeat this step to load all data 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)
```

Question 1 - Sentiment

The first question is about extracting sentiment or polarity of each tweet in my dataset. User firstly runs script `2.1 Derive the sentiment`. It load all the information about searched tweets using this command.

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

In the command we asked about searching the Twitter with our query I have mentioned earlier. We searched only for tweets in English with given filtering (date and max count). Then I inform the user

that the program is computing because for larger number of tweets it can take some time. The next we want to do is data cleaning so we will have better final results of polarity. We have defined method below which 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.

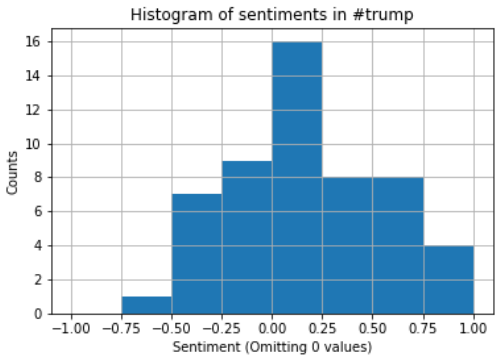
```
sentim_obj = []
for tweet in clear_tweets:
    sentim_obj.append(TextBlob(tweet))
```

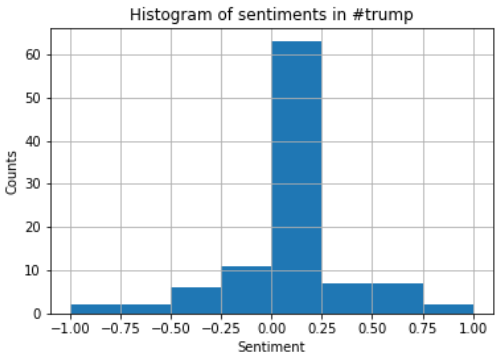
Then we make the similar thing but we create an array of pairs (text value of tweet and its sentiment value)

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

After that we create pandas `dataFrame` from this array of pairs for easier access. Now we can display data visually using histogram. In order to display histogram correctly without distortion I have decide to filter out all undecided sentiments. This gives us better visual representation of actual polarity of tweets. I have provide a quick look how would it looks like if I skipped this step as you can see below on the right side. 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. Then we just simply save more detailed data of sentiments including zeros values using pandas method `to_csv`, where we do not want index as column and 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()`.

```
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])
plt.savefig("01_sentiment_vis.png")
plt.show()
...
sentiment_df.to_csv("01_sentiment_res.csv",index=False,sep=";")
display(sentiment_df)
```





Question 2 – Hashtags and users

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. In the beginning of script `2.2 Top 10 hashtags and users` we make the same query as in previous script because I find it more reliable than making only one query in `2.1 Derive the sentiment`. It seems not working properly if I have done it only once so I have decide to repeat the same query which might give us the same dataset as before. I also inform user that computing have just begun.

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. To get list of all hashtags in one tweet I am using the method `tweet.entities.get("hashtags")`. After getting this list 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 transfer all the hashtags to lower case before proceeding.

```
hashtags_dict={}
for tweet in tweets:
    hashtags = tweet.entities.get("hashtags")
    for hashtag in hashtags:
        tag = hashtag['text'].lower()
        if tag in hashtags_dict.keys():
            hashtags_dict[tag] += 1
        else:
            hashtags_dict[tag] = 1
```

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

```
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 ten hashtags and put them to new dictionary. After that I simply creating 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` as I have already mentioned. Right after that I inform user again that the new computing have just started.

When it comes to second part of the question about fetching Top 10 users I will only makes some point on 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 @user 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 on will map user as a key to its real name. For accessing those values I will use `tweet.author.*` syntax from Twitter API.

```
users_dict={} #user->counts
names_dict={} #user->name
...
    user = tweet.author.screen_name
    name = tweet.author.name
...
        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. Program inform user about finishing by stating “Done!”

```
df_users = pd.DataFrame(res_users,
columns=["@user","real_name","occurrence"])
...
df_users.to_csv('02_top10_users.csv',index=False,sep=";")
display(df_users)
```

I have included Python source file and samples of CSV and PNG files while submitting this write up to give better view of the whole program. For samples I have used following inputs:

```
Please enter the event: #trump
Enter the oldest possible date of tweets. Use format "YYYY-MM-DD": 2021-01-
01
Enter the maximum number of tweets: 100
```