Gathering Twitter Data Using APIs

1. Introduction and Preparation

Twitter data can be gathered using an API (Application Programming Interface). For details on an API, please refer to a Wikipedia page (https://en.wikipedia.org/wiki/API).

To access Twitter data, users must obtain permission from Twitter, regardless of the tools they choose to use. Please go to the link below to find more information about steps for obtaining permission.

https://developer.twitter.com/en/docs/twitter-api/getting-started/guide

Note that the permission is not always guaranteed due to the internal review policies and practices at Twitter.

Once you successfully open a developer account and obtain a set of API keys (a.k.a. Consumer Keys) with a set of Access Tokens, you can now choose tools (e.g., Python, R, etc.) for gathering Twitter data.

For the purpose of demonstration, this course uses a Python package (twitter) because it can be used without any restrictions on use and modification.

For those who are not familiar with Python, please refer to the documentation at Python.org for installation (https://www.python.org/downloads/) and a tutorial (https://docs.python.org/3/tutorial/).

Note that this course does not provide any training or material on Python for beginners. If you prefer a non-programming tool for Twitter data collection, a few Graphical User Interface tools can be helpful. Below are some of the GUI tools for consideration.

NodeXL Pro: https://nodexl.com/

Orange Data Mining: https://orangedatamining.com/

Please remember that they also require you to obtain a set of API keys and a set of Access Tokens before you use them. In addition, you may need to pay fees (NodeXL Pro, after a free trial).

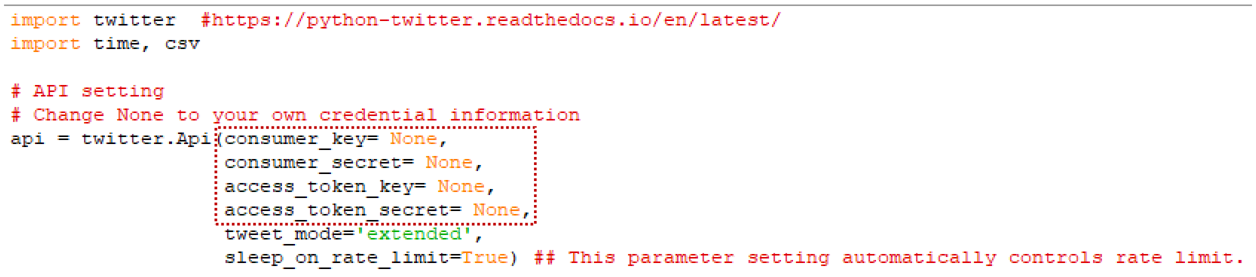
2. Use Python for Twitter Data Collection

If you want to try the Python package (twitter), please install the package first by following the instructions at https://python-twitter.readthedocs.io/en/latest/installation.html.

If the package is installed properly, put a Python script file (collect_twitter_demo.py) provided by this course into a folder you want to use.

Go to the folder where the code file is stored and open the code file with a text editor.

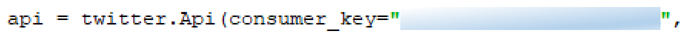
Please find the code lines as shown below (in a dotted box) and replace each 'None' with keys and tokens you obtained from Twitter. You must provide information about 'consumer_key', 'consumer_secret', 'access_token_key', and 'access_token_secret' to run the code.



```
import twitter  #https://python-twitter.readthedocs.io/en/latest/
import time, csv

# API setting
# Change None to your own credential information
api = twitter.Api(consumer_key= None,
                  consumer_secret= None,
                  access_token_key= None,
                  access_token_secret= None,
                  tweet_mode='extended',
                  sleep_on_rate_limit=True) ## This parameter setting automatically controls rate limit.
```

Each key and secret are combinations of numbers and alphabetical characters. When typing the key and secret information, don't forget to put quotation marks at the beginning and end of each piece of information as follows.

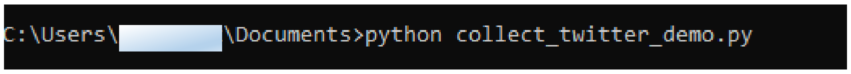


```
api = twitter.Api(consumer_key="[redacted]",
```

Once you finish typing all keys and secret information, save the code file and close it.

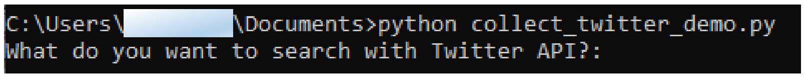
Then, run the code on a prompt as follows. Make sure that you are on the path of the folder (C:\Users\ ... \Documents) where code is saved.

First, type a command ("python collect_twitter_demo.py") and click the Enter key.



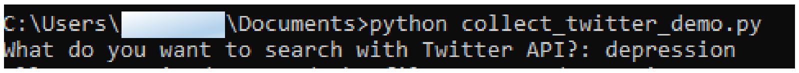
```
C:\Users\[redacted]\Documents>python collect_twitter_demo.py
```

Then, you will see a question "What do you want to search with Twitter API?:" on the prompt as follows.



```
C:\Users\[redacted]\Documents>python collect_twitter_demo.py
What do you want to search with Twitter API?:
```

Please type a word of your interest to search in tweets. For example, this demo uses 'depression' for a search term.

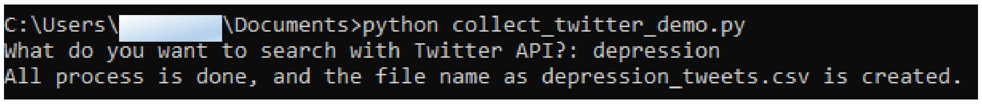


```
C:\Users\[redacted]\Documents>python collect_twitter_demo.py
What do you want to search with Twitter API?: depression
```

You can use operators to search two or more words in tweets. For example, 'snow day' will match tweets containing the keywords 'snow' and 'day' while 'snow OR day' will match tweets containing at least one of 'snow' and 'day.' If you want to query a hashtag, you need to attach a sharp sign (#) before a search term (e.g., #NoSchool). For more information about operator syntax, see the documentation at https://developer.twitter.com/en/docs/labs/recent-search/guides/search-queries.

Once you type the search word ('depression') and click the Enter key, the code will implement the API call procedure.

If the code is implemented successfully and gathers data from Twitter, it will produce a message "All process is done, and the file name as depression_tweets.csv is created." on the prompt.



```
C:\Users\[redacted]\Documents>python collect_twitter_demo.py
What do you want to search with Twitter API?: depression
All process is done, and the file name as depression_tweets.csv is created.
```

You can find the output file in the folder where the code is saved.

The output file contains a variety of information associated with gathered tweets.

a. Account ID

b. Account description

c. Count of favorites

d. Count of followers

e. Count of friends

f. Account name

g. Account profile image URL

h. Tweet id

i. Tweet text

j. Place (city-level)

For details about each piece of information can be found at https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/overview.

3. Code Overview

The code file used in the demonstration consists of five parts.

a. Importing modules: This part prepares the use of the core Python package 'twitter' and printing results in csv format.

```
import twitter  #https://python-twitter.readthedocs.io/en/latest/
import time, csv
```

b. Preparing API setup: Users plug in information of keys and tokens to enable the code to access Twitter data.

```
# API setting
# Change None to your own credential information
api = twitter.Api(consumer_key= None,
                  consumer_secret= None,
                  access_token_key= None,
                  access_token_secret= None,
                  tweet_mode='extended',
                  sleep_on_rate_limit=True) ## This parameter setting automatically controls rate limit.
```

Details about parameters (e.g., sleep_on_rate_limit) and options (e.g., True or False) are provided in the documentation at https://python-twitter.readthedocs.io/en/latest/rate_limits.html

c. Importing Twitter information: This part defines what information is gathered from the accessed Twitter data. Some code lines come with auxiliary information for parameters and options. In the code, for example, only tweets in English are collected (status.lang = “en”).

```
def get_tweets (status, api, csvwriter):
    # a list to save tweet attributes
    temp = []
    tweet_id = status.id
    user_id = status.user.id
    # sleep for controling rate limit
    time.sleep(0.1)
    if (status.lang == "en"): ## collect tweets only in english
        user_info = api.GetUser(user_id)
        account_description = user_info.description.replace("\r\n", " ").replace("\n", " ").replace("\r", " ")
        account_favourites_count = user_info.favourites_count
        account_followers_count = user_info.followers_count
        account_friends_count = user_info.friends_count
        account_name = user_info.name.replace("\r\n", " ").replace("\n", " ").replace("\r", " ")
        account_profile_image_url = user_info.profile_image_url
        account_screen_name = user_info.screen_name.replace("\r\n", " ").replace("\n", " ").replace("\r", " ")
        # tweets is trucated according to the new policy
        # https://github.com/bear/python-twitter/issues/420
        # https://stackoverflow.com/questions/38717816/twitter-api-text-field-value-is-truncated
        # We should use full_text instead of text
        # For retweets we need one more steps too.
        if status.retweeted_status == None :
            tweet_text = status.full_text.replace("\r\n", " ").replace("\n", " ").replace("\r", " ")
        else:
            temp_text = status.full_text
            new_text = status.retweeted_status.full_text
            ## find the overraped part and replace with the full text
            ## There are cases where the tweets are very short.
            if len(new_text) >= 3:
                length_compare = round(len(new_text) / 3)
                temp_text = temp_text.split(new_text[:length_compare])
                final_text = temp_text[0] + new_text
                tweet_text = final_text.replace("\r\n", " ").replace("\n", " ").replace("\r", " ")
            else:
                tweet_text = status.full_text.replace("\r\n", " ").replace("\n", " ").replace("\r", " ")
        temp.append(user_id)
        temp.append(account_description)
        temp.append(account_favourites_count)
        temp.append(account_followers_count)
        temp.append(account_friends_count)
        temp.append(account_name)
        temp.append(account_profile_image_url)
        temp.append(account_screen_name)
        temp.append(tweet_id)
        temp.append(tweet_text)
        ## There are three types of geostatus: geo, coordinates, and place.
        if status.place != None:
            if status.place["full_name"] != None:
                temp.append(status.place["full_name"])
            else:
                temp.append("N/A")
        else:
            temp.append("N/A")
        csvwriter.writerow(temp)
    return csvwriter
```

d. Prompting users to input a search term: This allows users to type a search term of interest.

```
## ask search term to user
search_term = input("What do you want to search with Twitter API?: ")
```

e. Writing results in a csv file: Collected data are recorded in a csv file and a completion message appears when the collection is finished.

```python
filename = search_term + "_tweets.csv"

with open(filename, 'a', encoding="utf-8", newline="") as out_file:
    ## create a list that will save tweet instances
    ## following are field names
    temp = []
    temp.append("account_id")
    temp.append("account_description")
    temp.append("account_favourites_count")
    temp.append("account_followers_count")
    temp.append("account_friends_count")
    temp.append("account_name")
    temp.append("account_profile_image_url")
    temp.append("account_screen_name")
    temp.append("tweet_id")
    temp.append("tweet_text")
    temp.append("place")
    CSVWriter = csv.writer(out_file)
    CSVWriter.writerow(temp)
    ## Return twitter search results for a given term. You must specify one of term, geocode, or raw_query.
    tweets = api.GetSearch(search_term, count=100, lang="en", include_entities=True )
    ## loop over the returned statuses to extract attributes needed
    for tweet in tweets:
        CSVWriter = get_tweets(tweet, api, CSVWriter)

print("All process is done, and the file name as "+ filename + " is created.")
```

The code is set to import up to 100 tweets but the number of imported tweets can be modified by changing the number at “count=100” in “tweets = api.GetSearch(search_term, count=100, lang=”en”, include_entities=True.” If you want to change the code to import other information such as follower ids and retweet count, refer to the documentation of the package ‘twitter’ at https://python-twitter.readthedocs.io/en/latest/index.html.