Data Collection - Tweets per User

BAMP 2022 - MCT 4 - Ante Jelavic, Franziskus Perkhofer, Manuel Mencher, Melissa Ewering, Tim Ritzheimer

Short description: This script is used to collect tweets using the Twitter API. The basis for this is the file "Demographics.xlsx" in which 100 Twitter user IDs and associated demographic data were collected manually. For these users, a 3 year history of all tweets will be retrieved (2019-2021).

This script was written based on the Twitter API documentation, partly whole sections were copied and adjusted where necessary. Link: https://developer.twitter.com/en/docs

Loading of all necessary packages

In [1]:
```
# For sending GET requests from the API
import requests
# For saving access tokens and for file management when creating and adding to the dataset
import os
# For dealing with json responses we receive from the API
import json
# For displaying the data after
import pandas as pd
# For saving the response data in CSV format
import csv
# For parsing the dates received from twitter in readable formats
import datetime
import dateutil.parser
import unicodedata
#To add wait time between requests
import time
```

The Bearer Token is an unique token that is obtained when successfully applying for access to the Twitter API, as this is private data, it was removed after the cell was executed.

In [2]:
```
bearer_token = 'To be filled'
os.environ['TOKEN'] = bearer_token
```

This function is used to create the specific search query / URL including all information about the specific request

In [3]:
```
def create_url():
    # Replace with user ID below
    #user_id = 813286
    return "https://api.twitter.com/2/users/{}/tweets".format(user_id)
```

Request details like the included attributes and the maximum amount of results are specified in the following function.

In [4]:
```
def get_params(pagination_token):
    # Tweet fields are adjustable.
    # Options include:
    # attachments, author_id, context_annotations,
    # conversation_id, created_at, entities, geo, id,
    # in_reply_to_user_id, lang, non_public_metrics, organic_metrics,
    # possibly_sensitive, promoted_metrics, public_metrics, referenced_tweets,
    # source, text, and withheld
    return {
        'max_results': 100,
        #"tweet.fields": "id,text,author_id,geo,conversation_id,created_at,lang,public_metrics,referenced_tweets,reply
_settings,source",
        #
        'expansions': 'author_id,in_reply_to_user_id,geo.place_id',
        'tweet.fields': 'id,text,author_id,in_reply_to_user_id,geo,conversation_id,created_at,lang,public_metrics,refe
renced_tweets,reply_settings,source',
        'user.fields': 'id,name,username,created_at,description,public_metrics,verified',
        'place.fields': 'full_name,id,country,country_code,geo,name,place_type',
        #
        'start_time':'2019-01-01T00:00:01Z', #START TIME selceted to get full year 2019 - 2021; 3 years
        'end_time':'2022-01-01T00:00:01Z',
        'pagination_token' : pagination_token
    }
```

The following two functions are used to establish the connection to the Twitter API

In [5]:
```
def bearer_oauth(r):
    """
    Method required by bearer token authentication.
    """

    r.headers["Authorization"] = f"Bearer {bearer_token}"
    r.headers["User-Agent"] = "v2UserTweetsPython"
    return r
```

In [6]:
```
def connect_to_endpoint(url, params):
    response = requests.request("GET", url, auth=bearer_oauth, params=params)
    print(response.status_code)
    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(
                response.status_code, response.text
            )
        )
    return response.json()
```

Sequence logic for the requests

In [7]:
```
# general skeleton
def retrieve_tweets ():

    df = pd.DataFrame()
    url = create_url()

    end_criterion = 0
    next_token = None # initialize for first run
    while end_criterion == 0:
      # perform web service call
      params = get_params(next_token)
      json_response = connect_to_endpoint(url, params)
      df_tmp = pd.DataFrame(json_response['data'])
      try:
        df = df.append(df_tmp)
        print("Log: df_tmp appended to df")
      except NameError:
        # should only appear the first time
        print("Log: First run, set df with df_tmp")
        df = df_tmp
      print("Log: df has now a length of " + str(len(df)))

      try:
        next_token = json_response['meta']['next_token']
      except KeyError:
        print("Log: Caught Error - No next token available.")
        next_token = None

      if not next_token:
        print("Log: No next token available, all tweets retrieved")
        end_criterion = 1
    #  if len(df) >= max_tweets:
    #    print("Log: Maximal number of " + str(max_tweets) + " tweets we want to collect reached.")
    #    end_criterion = 1
      if end_criterion == 0:
        # only wait if the loop continues
        time.sleep(3) # Sleep for 3 seconds
    return df
```

Applying the sequence logic in a for loop to retrieve the tweets for all 100 user ids (Exemplary output for only 2 user ids)

In [8]:
```
df_output = pd.DataFrame()

df_input = pd.read_excel('Demographics.xlsx') #Needs to be saved in "Scripts" folder for runtime

counter = 1
for x in df_input["ID"]:
    user_id = x
    df_output = df_output.append(retrieve_tweets())
    print("Completed: " + str(counter))
    counter = counter + 1
```

```
200
Log: df_tmp appended to df
Log: df has now a length of 99
200
Log: df_tmp appended to df
Log: df has now a length of 199
200
Log: df_tmp appended to df
Log: df has now a length of 286
Log: Caught Error - No next token available.
Log: No next token available, all tweets retrieved
Completed: 1
200
Log: df_tmp appended to df
Log: df has now a length of 100
200
Log: df_tmp appended to df
Log: df has now a length of 200
200
Log: df_tmp appended to df
Log: df has now a length of 300
200
Log: df_tmp appended to df
Log: df has now a length of 400
200
Log: df_tmp appended to df
Log: df has now a length of 500
200
Log: df_tmp appended to df
Log: df has now a length of 600
200
Log: df_tmp appended to df
Log: df has now a length of 700
200
Log: df_tmp appended to df
Log: df has now a length of 715
Log: Caught Error - No next token available.
Log: No next token available, all tweets retrieved
Completed: 2
```

Saving the output to csv

In [9]:
```
df_output = df_output.replace('\n', '', regex = True)
df_output.to_csv("Tweets.csv") # File is saved in the "Scripts" folder and will be afterwards moved to output folder
```