Reddit Web Scrape

October 24, 2022

[268]:
```
#https://github.com/c7blackjack/Sentiment-Analysis

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import pandas as pd
import numpy as np
import os
import requests.auth
from rich import print

# Import file containing all access credentials
import reddit_creds as rc
```

[47]:
```
# Importing credentials from file not included to keep passwords and access keys
# private in public access github
CLIENT_ID = rc.CLIENT_ID
CLIENT_SECRET = rc.CLIENT_SECRET
USERNAME = rc.USERNAME
PASSWORD = rc.PASSWORD
```

[194]:
```
# This process authenticates a user in the process of connecting
# to the Reddit API.
client_auth = requests.auth.HTTPBasicAuth(CLIENT_ID, CLIENT_SECRET)

#The post data is what is sent to Reddit to gain access to the API in order to␣
 ↪pull data
post_data = {'grant_type': 'password', 'username': USERNAME, 'password':
 ↪PASSWORD}
```

[195]:
```
# The headers parameter is what is used to put in personalized information into␣
 ↪the
# pulled data
headers = {
    'User-Agent': "A red scrapper"
}
```

[196]:
```
# Token Access ID is where we tell Python to look for the connection point to␣
↪the
# Reddit API
TOKEN_ACCESS_ENDPOINT = "https://www.reddit.com/api/v1/access_token"

# This is our first resonse that I created as a way to check whether the␣
↪connection
# to the API was successful or not. It will return a 401 Error and if successful
# it will return a 200 message, in which case we save our token_id to continue␣
↪to pull
# and access the API and its data.
response = requests.post(TOKEN_ACCESS_ENDPOINT,data=post_data,headers=headers,␣
↪auth = client_auth)
if response.status_code == 200:
    token_id = response.json()['access_token']
```

[197]:
```
# The OATH_ENDPOINT is the base domain that we want python to use so that we␣
↪will
# attach subreddit domains to to pull using a list
OAUTH_ENDPOINT = 'https://oauth.reddit.com'

# Reddit has a max post pull of 100 posts per request so I set the limit here␣
↪to the
# max later we will incorporate another parameter named 'after' which give the␣
↪id
# of the last post of the 100 that were most recently pulled. Using this␣
↪information
# the tool will base its request off any posts older than that 'after' ID.
params_get = {
    'limit' : 100
}


headers_get = {
    'User-Agent': "A red scrapper",
    'Authorization': "Bearer " + token_id,
}
```

[199]:
```
## Testing whether the connection to Reddit is good.
response = requests.get(OAUTH_ENDPOINT + '/r/personalfinance/',␣
↪headers=headers_get, params = params_get)
print(response)
```

```
<Response [200]>
```

```
[280]: ## Function that returns roughly 1000 max subreddit posts

def get_subs(subreddit):
    temp_df = pd.DataFrame()
    i=0
    params_get = {
        'limit' : 100
    }
    for i in range(0,10):
        response = requests.get(OAUTH_ENDPOINT + subreddit,␣
 ↪headers=headers_get, params = params_get)
        data = response.json()['data']['children']
        after_key = response.json()['data']['after']
        before_key = response.json()['data']['before']
        for post in data:
            temp_df = temp_df.append({
                'subreddit': post['data']['subreddit'],
                'text': post['data']['selftext']
            }, ignore_index = True)
        params_get = {
            'limit' : 100,
            'after': after_key
        }
        i += 1
    return temp_df
```

```
[298]: ####Tool to check makeup of text in subreddit, subs like sports have little to␣
 ↪no textual content

temp = get_subs('/r/poems/')
print('Percent empty strings: ', round((temp['text'].values == '').sum()/
 ↪len(temp),2)*100, '%')
```

```
Percent empty strings:  3.0 %
```

```
[299]: # This is a list of subreddits that we want to pull posts from
subs = ['/r/personalfinance/',
        '/r/personaltraining/',
        '/r/poems/',
        '/r/keepwriting/',
        '/r/story/']
```

```
[300]: df = pd.DataFrame()
```

```
[301]: df_list = []
for sub in subs:
    temp_df = get_subs(sub)
    df_list.append(temp_df)
df = pd.concat(df_list,axis=0)
```

[314]:
```
print('There are ', len(df), ' records.')
print('Percent empty strings: ', (round((df['text'].values == '').sum()/
 ↪len(df),2))*100, '%')
```

```
There are  4773  records.


Percent empty strings:  7.000000000000001 %
```

[315]:
```
df.to_excel("reddit-data.xlsx",
             sheet_name='Reddit Data')
```

```
<ipython-input-315-2f5fb2ed45e8>:1: UserWarning: Pandas requires version '1.4.3'
or newer of 'xlsxwriter' (version '1.3.7' currently installed).
  df.to_excel("reddit-data.xlsx",
```