Twitter REST APIs

This is the unique APIs provided by twitter as programmatic access to read and write Twitter dara. Author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

About rate limit, it divides the rate limit window into 15 minute chunks per endpoint, with most individual calls allowing for 15 requests in each window. Here I will implement how to get the profile image of one user' s followers. After get all id numbers, there can be two ways to get their images: first one is directly get the image through API request; the other one is get the whole profiles through API then extract the images. By comparing time limits of these two ways, I choose the second way. Next, get_id will get all followers' id; get_user will use id to lookup all followers' profile; get_image will extract image from the profile.

How to use - Tweepy

Before we get started to program through the API, we have to create a Twitter App. First, you need to have a twitter account; Second, visit the apps.twitter.com and sign in your account if necessary to create an App. Third, enter your App Name, Description and your website address. You can leave the callback URL empty. Fourth, accept the TOS, and solve the CAPTCHA. Fifth, submit your form by clicking the Create you Twitter Application. Finally, you will get your keys: consumer_key, consumer_secret, access_token, access_token_secret.

I make use of the Tweepy for programming with Python. If you would like to get started with Tweepy. The tutorial wesite is recommended:

docs.tweepy.org/en/v3.5.0/getting_started.html

Install Tweepy from Github:

pip install tweepy

More can be found on github.com/tweepy/tweepy

Note: Timeout Problem is somewhat disturbing for scrawlers.

Get ID

This paragraph of codes is necessary for connecting with API.

```
import sys
import tweepy
import time
from random import randint

# keys preparations
CONSUMER_KEY = 'YOUR CONSUMER KEY'
CONSUMER_SECRET = 'YOUR CONSUMER SECRET'
ACCESS_TOKEN = 'YOUR ACCESS TOKEN'
ACCESS_TOKEN_SECRET = 'YOUR ACCESS TOKEN SECRET'

try:
    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
    api = tweepy.API(auth, wait_on_rate_limit=True,
wait_on_rate_limit_notify=True, compression=True)
```

Then we will use cursor here in case of timeout error. In this way, even if there is a time out error when we run the program, we can know from the cursor where we should restart then. Notice the followers are keeping changing, but the cursor will not change.

```
    c = tweepy.Cursor(api.followers_ids, id = 25073877,
cursor=-1) # Donald Trump
    print "type(c)=", type(c)
    print c.iterator.next_cursor
```

Here the example is Donald Trump’ s ID. By visiting the gettwitterid.com, you can get anyone’ s twitter id. Then we will limit our request frequency and deal with the response.

```
    t = 0
    for page in c.pages():
        file =“ids_”+ str(t)
        fo = open(file,”w”)
        tmp = []
        for i in range(0, len(page)):
            tmp.append(str(page[i])+”\n”)
        fo.writelines(tmp)
        print c.iterator.next_cursor
        fo.close()
        t += 1
        if (t % 15 == 0):
            time.sleep(200)
```

Get PROFILE (lookup)

Since the response is in form JSON, we could use pickle to deal with our profile output. Connection with Twitter REST API is same as the last part, so we simple skip it. And also the input files are one id per line, so read file is easy and skipped here too. From get_id, we will get each file with 5000 ids, and now since lookup can only ask for 100 profiles each time, so we will have 50 pkls for each id file.

```
temp = []
    t = 0
    c = 0
    for line in fin.readlines():
        if (len(line)!=0):
            temp.append(line.strip())
            t += 1
            if (t == 100):
                print ','.join(temp)
```

```
            user = api.lookup_users(user_ids=[','.join(temp)])
                fout = open('test/test_' + fx + '_' +
str(c).zfill(6) + '.pkl','w')
                c += 1
                pickle.dump(user, fout)
                fout.close()

                temp = []
                t = 0
    fin.close()
```

Extract IMAGE

This part has nothing to do with the REST API. We will extract what we need from the pkl files (JSON). Therefore we will use the urllib. (import urllib)

```
for i in range (0,100):
    s =
str(page[i].profile_image_url_https).replace("_normal.",".")
    form = ""
    flag = 0
    if (s[len(s)-2]=="e"):
        form = ".jpeg"
    else: form = ".jpg"
    id = page[i].id
    print form
    image = name + "_" + index + "_" + str(i).zfill(2) + "_"
+ str(id) + form
    image = "test/" + image
    if (s[8]=="p"):  # remove default images
        urllib.urlretrieve(s,image)
```

Notice default profile images always have common '_normal', by removing those urls we can simply remove all default profile images.

Sample IDs

If you want to sample a part of the ids given number n, then we can do like this:

1. Make sure about total id number m
2. Write 1 to m into list.txt
3. Read in list.txt in a list, all are available indexes
4. Sample n indexes, and
   write indexes into samp_index.txt,
   write ids into samp_id.txt
5. Rewrite available ids into list.txt

Here I use random.randrange(0, n, 1), you can use a better random method. The codes are long but simple, you can refer them in my github~

Extract Location and Check (US) by Google Map API

First extract the locations with ids, this is same as before (page[i].location). Then we will get lists of locations, but they are not formal. No standard formula can we apply to them. You may get the location lists like this:

```
56743913    Little Rock Arkansas
50764967    slc, ut
357351262    Close to u!!!!!!!!
```

Even if there is a location, it can be not very programming-friendly. After checking and comparing, we find that the Google Map API is more available however the limit is somewhat disturbing.

Google Map API (geocoding)

You will need a gmail account to use the API. Then you can get a key by creating geocoding port. That is similar to twitter API. Here set default encoding as utf8, so we can make the output content understandable by us. The codes below are not preprocessed. Actually I implemented a preprocessed version, it can save about 0.1, and it still cannot fulfill our needs.

```
f_location = open('loc', 'a')
   for i in range (0,100):
      id = str(page[i].id)
      if (page[i].location):
         s = str(page[i].location.encode('utf-8'))
         geocode_result = gmaps.geocode(s)
         if (geocode_result):
            result = geocode_result[0]['address_components']
            num = len(result)
            if (num >= 3):
               result1 = result[-3]['long_name']
               result2 = result[-2]['long_name']
               result3 = result[-1]['long_name']
               print result1+'   '+result2+'   '+result3
               if (result1=='United States' or
result2=='United States' or result3=='United States'):
                  f_location.write(id+'  '+s+'
'+result1+'   '+result2+'   '+result3+'  '+'true'+'\n')
               else:
                  f_location.write(id+'  '+s+'false'+'\n')

   f_location.close()
```

Count number of Common Followers

In this part, we will implement to solve the total number of common followers of Hillary and Trump. By simply comparing the ids we can finally get our goal number. By set operation &, we can find common ids, so I will read the ids into two sets: one for Hillary, one for Trump. Then we can get a number. The codes can be paralled, but I have not try that.