THE UNIVERSITY OF SYDNEY

COMP5313 Lab 3

Twitter social interactions

The goal of this lab is to use the twitter public API in python to study social interactions. In this lab we will start using the Twitter 1.15.0 module for python. Note that the number of requests allowed per 15 minutes per token is limited as explained at https://dev.twitter.com/rest/public/rate-limiting.

Exercise 1: Registering an application

We will start by registering an application on Twitter. This application allows us to authenticate ourselves before using the Twitter API (through for example a python application with the Twitter module).

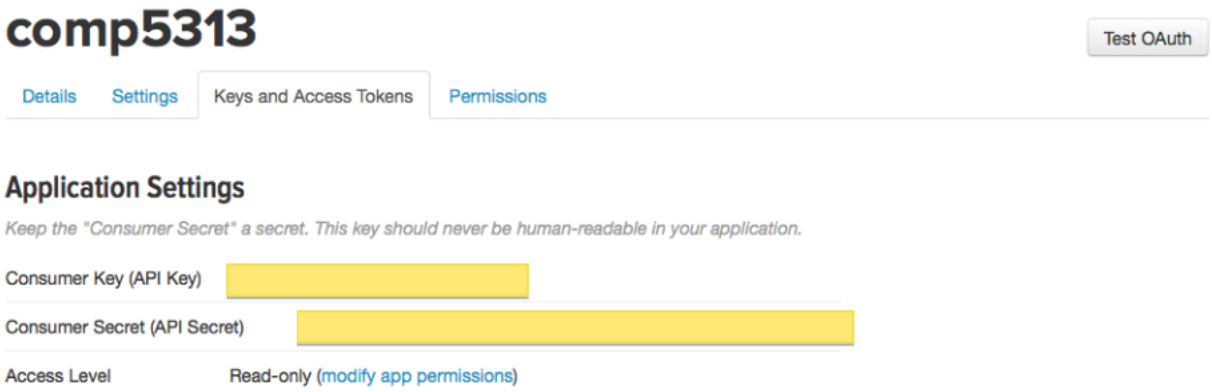

Figure 1: The application keys needed for your python program to authenticate itself should appear in the yellow rectangles

Go to https://apps.twitter.com. Create a twitter account if you do not have one and create a new application by clicking the "Create New App" button and filling some information like below:

- Name: COMP5313
- Description: Testing the twitter API
- Website: http://poseidon.it.usyd.edu.au/~gramoli/largescalenetworks/php/comp5313.php

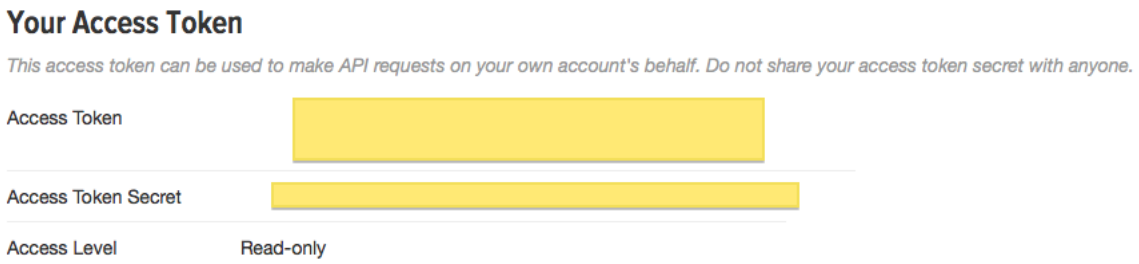

Figure 2: The token keys needed for your python program to authenticate itself should appear in the yellow rectangles

The important information needed from the application are the application keys that can be found under the "Keys and Access Tokens" tab (cf. Figure 1). You will have to generate the token that will appear below the application keys (cf. Figure 2).

Duration: 5 min

Exercise 2: Authentication through Python

Write a piece of python code in a file called **`authentication.py`** that can use the twitter API after authenticating itself. Make sure to import the **`twitter`** module. Let us then define a list of accounts containing the application account you have created above. This account **`app`** is defined by a tuple of key-value pairs where the values should indicate the application and token keys you generated in the previous question. The command **`.oauth.OAuth`** is used to authenticate your application with the information of account **`app`**. Finally the API can be invoked on the object returned by **`.Twitter(auth=auth)`**

The command **`twitter.Twitter(auth=auth)`** is used to authenticate

```
import twitter

accounts = {
 "comp5313" : { 'api_key' : 'API_KEY',
                'api_secret' : 'API_SECRET',
                'token' : 'TOKEN_KEY',
                'token_secret' : 'TOKEN_SECRET'
              },
}
app = accounts["comp5313"]

auth = twitter.oauth.OAuth(app["token"],
                           app["token_secret"],
                           app["api_key"],
                           app["api_secret"])
twitter_api = twitter.Twitter(auth=auth)
```

Duration: 10 min

Exercise 3: Followers and friends

Copy your file **authentication.py** into a new file called **friendship.py**.

```
cp authentication.py friendships.py
```

The *followers* of a user are the users who receive her tweets. The *friends* of a user are the users from whom she receives the tweets. One can easily track the number of followers/friends of a user, provided that this information is public by using **.followers.ids** described at https://dev.twitter.com/rest/reference/get/followers/ids. Append the following function that computes the number of followers to the code that you have written previously. Note that **cursor=0** indicates that there are no more results and that **.followers.ids(screen_name=screen_name, cursor=cursor)** returns the next page of results.

```
def getfollowernum(screen_name):
    cursor = -1
    followers = []
    while cursor!=0:
        result = twitter_api.followers.ids(screen_name=screen_name, cursor=cursor)
        followers += result["ids"]
        cursor = result["next_cursor"]
    return len(followers)
```

You can test the result of this function by invoking it on specific **screen_name**, the login username of a twitter user but not her real name (e.g., "marissamayer" for the CEO of Yahoo!). Note that you can write the same function to get the number of friends by using function **.friends.ids** described at https://dev.twitter.com/rest/reference/get/friends/ids.

How would you change this function and how would you use the resulting function to obtain a large-scale social network?

Duration: 15 min

Exercise 4: Extracting graph properties

Here we will use the **networkx** python module to observe the degrees of interaction of a Twitter user in a directed graph **DiGraph**. To this end we will use **.statuses.user_timeline()** that returns a set of tweets posted by a single user. As it only returns a maximum of 200 tweets in each call, we will invoke it multiple times to get more tweets from the same users (up to the maximum of 3200). First, copy your file **authentication.py** into a new file called **relations.py**.

```
cp authentication.py relations.py
```

Then, import the **networkx** module as follows:

```
import networkx as NX
```

Finally, append the following code to the file. Note the arguments **`count=200`** that indicate the number of tweets to return in each call, **`trim_user='true'`** to not include the user information and save bandwidth and processing time and **`include_rts="true"`** to include the retweets by this user, **`max_id`**, which represents the maximum tweet id to be considered, is set to the last **`max_id`** fetched-1 in each iteration.

```
screen_name="marissamayer"

args = {"count" : 200, "trim_user": "true", "include_rts": "true"}
tweets = twitter_api.statuses.user_timeline(screen_name = screen_name, **args)
tweets_new = tweets

while len(tweets_new) > 0:
    max_id = tweets[-1]["id"] - 1
    tweets_new = twitter_api.statuses.user_timeline(screen_name = screen_name,
        max_id=max_id, **args)
    tweets += tweets_new

user = tweets[0]["user"]["id"]
G = NX.DiGraph()

for tweet in tweets:
    if "retweeted_status" in tweet:
        G.add_edge(user, tweet["retweeted_status"]["user"]["id"])
    elif tweet["in_reply_to_user_id"]:
        G.add_edge(tweet["in_reply_to_user_id"], user)

print "Graph has ", G.number_of_nodes(), " nodes,",\
    G.number_of_edges(), " edges, and the maximum degree is ",\
max(G.degree().values())
```

What do you observe?

Note that you can export a graph into GraphML format to be imported in other graph visualisation tools, like Gephi, using the function **`NX.write_gexf(G, "mayer.gexf")`**.

Duration: 20 min