2017

Assignment 6

ROSS REELACHART

PROFESSOR MICHAEL L. NELSON

Question 1

Initially, I thought gathering Twitter followers and the links between them would be a task very similar to a previous assignment. However, I found that gathering the links between users was proving to be a more difficult with that particular method. So I needed to actually learn how to use Tweepy[1] to gather the information I needed. At least I could still use the keys and tokens I generated for the previous assignment.

NOTE: I was initially alerted to the troubles regarding the gathering of all the links by a post on the class discussion board. After I also found myself running into the "rate limit" error and not being able to find a sufficient workaround that I could implement, I settled with limiting myself to what I could get. Thus, I found the very small Twitter account "@ODUFootballCamp[2]" that only had ~60 followers at the time of data gathering. But even then, when attempting to find all the links between all the followers and the source account, I ran into the "rate limit" error. That is why, despite having approximately 60 followers linked to the source account, I only have a paltry two connections between the followers themselves. This is seen very obviously in the graph.

Similar to previous assignments with many steps that created many possible failure points, I chose to create a bunch of Python programs that attempted to complete one step of the process. The first step was gathering the initial accounts, or "nodes," connected to @ODUFootballCamp. This was completed using the *getTwitterNodes.py* program, which created the *twitterNodes* file. Then the links between the gathered nodes was found using *getTwitterLinks.py*, which created the *twitterLinks* file.

```
import tweepy
import sys
import json
import time

CONSUMER_KEY = "Vvp6YekJ62nw1SCvx9xNeBUWr"
CONSUMER_SECRET = "2P2bP3i8Ne3kgkFIgFBya3mRHbZgM2MvLTkv1GEDuxMywgPbGw"
OAUTH_TOKEN = "704338593108484096-DFScjuC1jzqT8b2U8rTYByvgPLJbkzi"
OAUTH_TOKEN_SECRET = "zrKA3L2astz0ycX1eX42qml5yhBNP4NqqxnqePXLKEbW1"

auth = tweepy.auth.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN, OAUTH_TOKEN_SECRET)
api = tweepy.API(auth)

f = open('twitterNodes','w')
def nodes():
    count = 1
    page_count= 0
    userData = []
    for user in tweepy.Cursor(api.followers, screen_name='ODUFootballCamp').items():
        usr = {}
        usr['screenName'] = user.screen_name
        usr['name'] =  user.name
        usr['id'] =  count
        usr['img'] =  user.profile_image_url
        usr['link'] = "https://twitter.com/"+user.screen_name
        usr['size'] =  40000
        page_count += 1
        userData.append(usr)
        count+= 1
    f.write(json.dumps(userData)+"\n")
    f.close()

nodes()
```

Figure 1 - Code for getTwitterNodes.py

```
import sys
import json
import time

f1 = open('twitterLinks','w')
read = open('twitterNodes','r')
def links():
    userData =[]
    for line in read:
        data= json.loads(line)
        for user in data:
            dict = {}
            dict['source'] = user['id']
            dict['target'] =  0
            userData.append(dict)
    f1.write(json.dumps(userData)+"\n")
    f1.close()

links()
```

Figure 2 - Code for getTwitterLinks.py

The next few steps were about putting the nodes and links together in a way that would give me the information necessary to make a nice D3 graph. So the program *checkSourceFollow.py* was used to find the followers of each follower that followed @ODUFootballCamp according to the *twitterNodes* file, and then create a file that was useable later for finding links. This produced the file *sourceTarget*.

```
import commands
import json
import time

f1=open('sourceTarget','w')

def getSource():
    read=open('twitterNodes','r')
    data= json.load(read)
    list= []
    for user in range(0,len(data)):
        sourceScreenName= data[user]["screenName"]
        for user1 in range(user,len(data)-1):
            targetScreenName = data[user1+1]["screenName"]
            checkSourceFollowersAndFollowing(sourceScreenName,targetScreenName)

def checkSourceFollowersAndFollowing(sourceScreenName,targetScreenName):
    dict= {}
    count = 0
    dict['source']= sourceScreenName
    dict['target']= targetScreenName
    f1.write(json.dumps(dict)+",\n")

getSource()
```

Figure 3 - Code for checkSourceFollow

Then using the *sourceTarget* file, the program *getAllLinks.py* was used to query the Twitter API and confirm if any of the possible links were between accounts connected to @ODUFootballCamp. This was where I encountered the "rate limit" error the most, so I was unable to find every possible link between all these accounts. This would later mean possibly even less true links that could be used. This produced the file *twitterLinksData*.

```
import tweepy
import commands
import json
import time

CONSUMER_KEY = "Vvp6YekJ62nw1SCvx9xNeBUWr"
CONSUMER_SECRET = "2P2bP3i8Ne3kgkFIgFBya3mRHbZgM2MvLTkv1GEDuxMywgPbGw"
OAUTH_TOKEN = "704338593108484096-DFScjuC1jzqT8b2U8rTYByvgPLJbkzi"
OAUTH_TOKEN_SECRET = "zrKA3L2astz0ycX1eX42qml5yhBNP4NqqxnqePXLKEbW1"

auth = tweepy.auth.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN, OAUTH_TOKEN_SECRET)
api = tweepy.API(auth)

f2 = open('twitterLinksData', 'w')

def getAllLinks():
    read = open('sourceTarget','r')
    data= json.load(read)
    for user in data:
        dict= {}
        sourceScreenName= user["source"]
        targetScreenName= user["target"]
        result = api.show_friendship(source_screen_name=sourceScreenName, target_screen_name=
        targetScreenName)
        dict['followed_by'] =result[0].followed_by
        dict['following'] =result[0].following
        dict['screen_name1']= result[0].screen_name
        dict['screen_name2']= result[1].screen_name
        f2.write(json.dumps(dict)+",\n")

getAllLinks()
```

Figure 4 - Code for getAllLinks.py

The program *getTwitterTrueLinks.py* used the *twitterLinksData* file to then find any real or "true" links between any of the accounts on my (rate limited) list. Because of my limited list, I only found two true connections, which were then placed in the file *twitterTrueLinks*.

```
import commands
import json
import time

def getTrueLinks():
    read = open('twitterLinksData','r')
    f2 = open('twitterTrueLinks','w')
    data= json.load(read)
    for user in data:
        dict = {}
        if user["following"] == True:
            dict['source']= user["screen_name1"]
            dict['target']= user["screen_name2"]
            f2.write(json.dumps(dict)+",\n")
        elif user["followed_by"]== True:
            dict['source']= user["screen_name2"]
            dict['target']= user["screen_name1"]
            f2.write(json.dumps(dict)+",\n")

getTrueLinks()
```

Figure 5 - Code for getTwitterTrueLinks.py

The final step regarding the data was putting it all together in format for use in D3. This was accomplished by the *makeGraphData.py* program. This produced the *graphIDs* file.

```
import commands
import json
import time

def passTrueLinksSourceAndTarget():
    read = open('twitterTrueLinks','r')
    data= json.load(read)
    for user in data:
        getIds(user["source"],user["target"])

def getIds(name1,name2):
    read = open('twitterNodes','r')
    f2 = open('graphIDs','a')
    data= json.load(read)
    for user in data:
        dict ={}
        if name1 == user["screenName"]:
            id = user["id"]
            dict['source'] = id
            f2.write(json.dumps(dict)+",\n")
        elif name2 == user["screenName"]:
            id = user["id"]
            dict['target'] = id
            f2.write(json.dumps(dict)+",\n")
    f2.close()

passTrueLinksSourceAndTarget()
```

Figure 6 - Code for makeGraphData.py

A step that I did manually was combining the data in *twitterNodes, twitterLinks, and graphIDs* into a single file called *finalD3Data.json*. This was ugly to look at so I used the Python "-m json.tool" function to make it look pretty (*prettyD3Data.json*).

A D3 code[3] in html was then written, based heavily off an existing[4] code[5] that was also on the hosting website. The resultant code and visualization can be found here:

https://bl.ocks.org/rreelachart/5d536965496668fe220afff0e59868c8

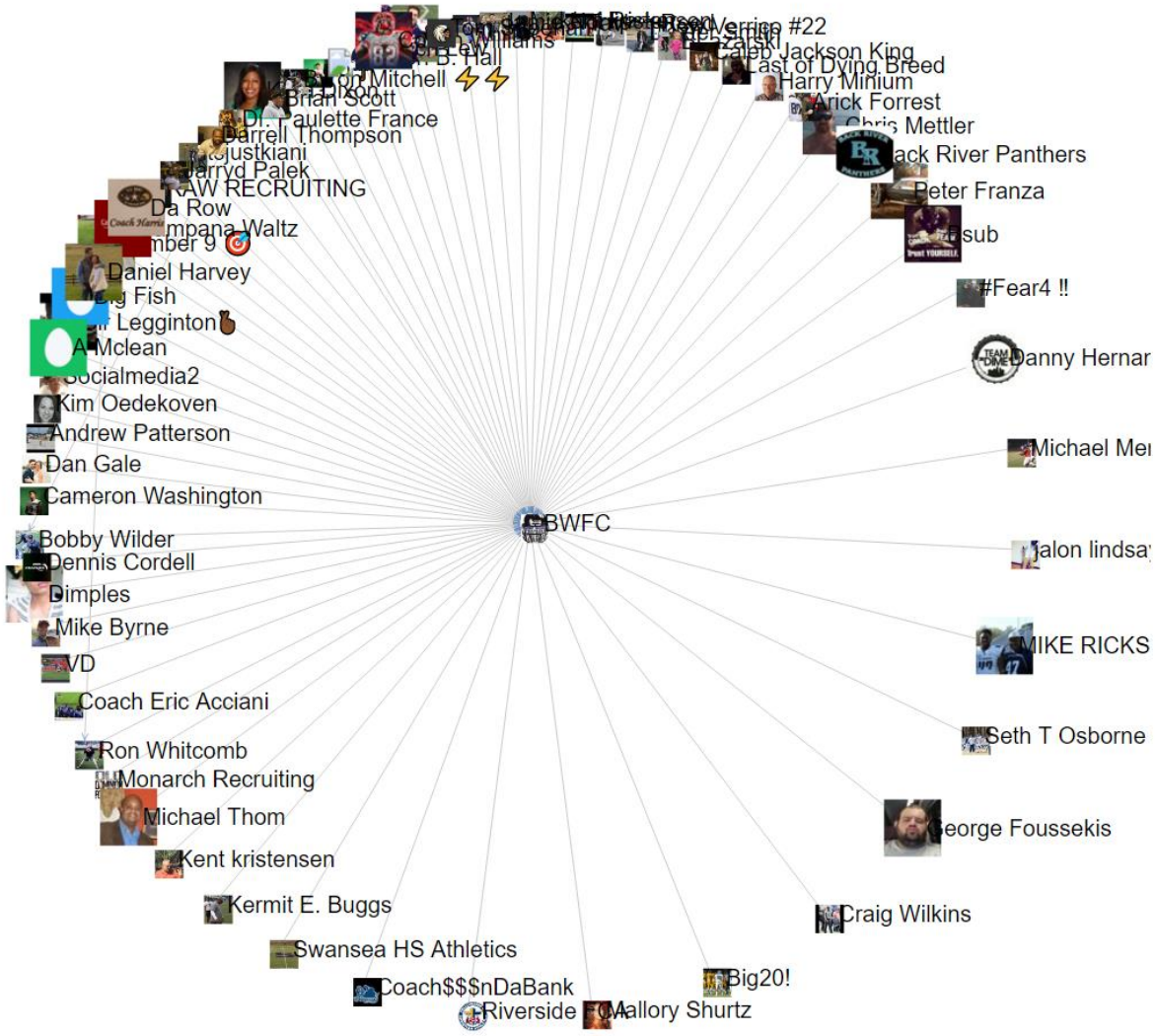

Figure 7 - The final graph. Note the meager two connects on the left side of the graph, due to "Rate limit" errors.

References

1.) http://tweepy.readthedocs.io/en/v3.5.0/api.html#user-methods
2.) https://twitter.com/ODUFootballCamp
3.) https://www.dashingd3js.com/d3js-first-steps
4.) http://bl.ocks.org/mbostock/1153292
5.) http://bl.ocks.org/eesur/be2abfb3155a38be4de4