Ryan Condotta

Final Report Assignment 6

Question 1:

In order to solve the problem of visualizing my twitter followers, I created the python file A6Q1.py and D3_Project files of index.html, style1.css, and Twitter1.js. In the A6Q1.py file, I began by getting screen names of my friends as shown in Figure 1. The process will get a user in my friends list and add the user's screen name to names list. After this process is done, it would output the information to 2 files of List.txt and List1.txt. The reason for them being in two separate list is so that I could compare the lists to see the relationships between each of my friends and myself to help determine the edges.

```
for user in tweepy.Cursor(api.friends).items(50):
        try:
                #print(count)
                print(user.screen_name)
                names.append(str(user.screen_name))
                process_status(user)
                count+=1
        except:
                pass
                count+=1
```

Figure 1

Once the two lists are formed of List1.txt and List.txt, I stored each of the lists of screen names into separate arrays so that I could iterate through each of the list so that each person would compare with everyone to determine if they follow each other or not. This is shown in Figure 2 by taking each screen name and plugging them into the show_friendship function which returns the relationship of if they follow each other or not. If they source was followed by the destination, then it would output the relationship to a file with the name of the source screen name.

```
for person in names:
        for peeps in names1:
                if person == peeps:
                        pass
                else:
                        relationship = api.show_friendship(source_screen_name=person,
target_screen_name=peeps)
                        source, destination = relationship
                        print(source.followed_by, destination.screen_name)
                        print(destination.followed_by, source.screen_name)
                        print("\n")
                        if source.followed_by == true:
                                filename =
r"C:\Users\Ryan\Documents\WebScience\Assignment6\Relationships\'"+source.screen_name + ".txt"
                                outfile = open(filename, 'a')
                                outfile.write(source.screen_name + " "+str(source.followed_by) +" "+
destination.screen_name+"\n")
                                outfile.close()
```

Figure 2

Once all of the relationships were solved for who followed who as to define the edges, a csv file was created with the source and destination named Source_targets.csv. The format of the file is shown in Figure 3. This

```
source,target
Ryan_Condotta,CNN
Ryan_Condotta,TheEPLVines
```

Figure 3

The last process was visualizing the relationships and creating the graph in D3. To accomplish this, the files index.html, style1.css, and Twitter1.js. In the file index.html, it basically creates the html for the web page that will store the style.css and Twitter1.js functionality and formatting. In the file style1.css, it creates the formatting for the page as well as formats for key elements in the Twitter.js file such as edges, nodes, and text. The main functionality for visualizing the graph happens in the Twitter1.js file, where it takes in a csv file and creates nodes and links as shown in Figure 4. Instead of labeling each edge, I made a directed graph with arrows for who follows who. This way it would not clutter the graph with text and make it more legible for the user.

```
d3.csv("Source_targets.csv", function(error, links) {

var nodes = {};

// Compute the distinct nodes from the links.
links.forEach(function(link) {
   link.source = nodes[link.source] ||
      (nodes[link.source] = {name: link.source});
   link.target = nodes[link.target] ||
      (nodes[link.target] = {name: link.target});
   //link.value = +link.value;
});
```

Figure 4

The rest of the file creates the individual pieces such as nodes, links, and arrows for visualizing in google chrome. It produces the graph as shown in Figure 5. The output needs to be in google chrome or else it will not display the arrows for the edges which is a huge part of the graph in order to determine who is following who. I couldn’t determine the reason why it will only appear in google chrome but as for now, the graph to its full functionality will only show all of its visualization and properties in google chrome.

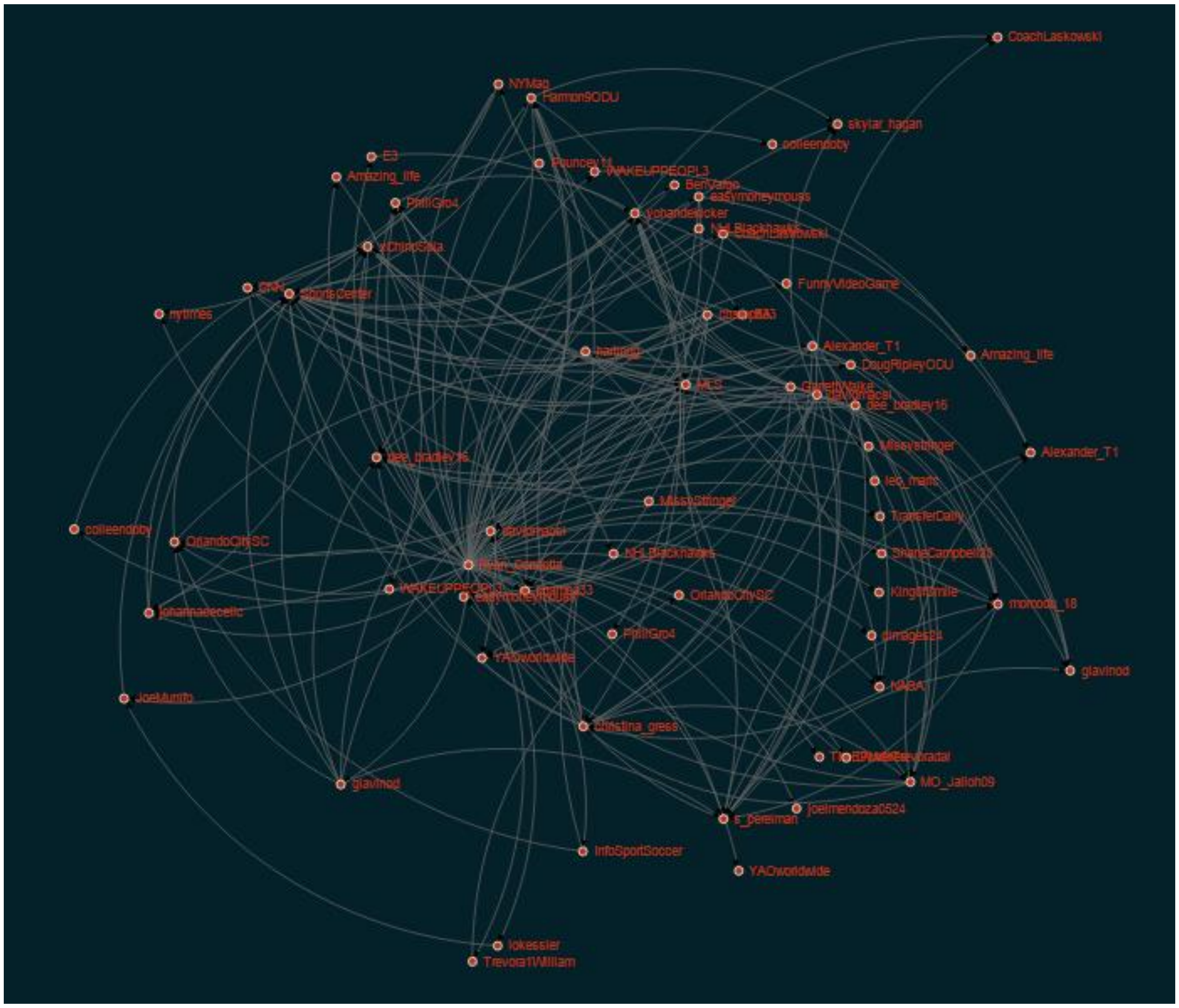

Figure 5

Question 2:

In order to determine gender homophilly for my graph in Figure 5, I first created the python file to determine the actual user’s name and not screen name, as well as, determine the user’s name gender based on the request from https://genderize.io/. The process is shown in Figure 6 which gets the user’s name first then sends a request to the web page with the person in order to determine if the gender for the name was male, female or null and stores it in an array.

```
#get users
for person in snames:
        user = api.get_user(person)
        anames.append(str(user.name))

#get gender
for person in snames:
        file = r"https://api.genderize.io/?name="+person
        try:
                response = urllib.request.urlopen(file)
                soup = BeautifulSoup(response, 'html.parser')
                for line in soup:
                        if 'female' in line:
                                print('female')
                                genders.append('female')
                        elif 'male' in line:
                                print('male')
                                genders.append('male')
                        elif 'null' in line:
                                print('null')
                                genders.append('null')
        except:
                print(person)
```

Figure 6

Once the names and genders are in designated lists, I outputted that information to an excel file named Gender_Table.xlxs where it store the gender and names in the first two columns. Figure 7 shows part of the names and gender table, as well as, all the names without genders that would be removed from the graph above.

Names	Gender
CNN	null
Live EPL Goals	female
William Trevor adair	male
NASA	null
New York Magazine	female
The New York Times	female
Gaming Humor	null
Moussa Diallo	male
Shane Campbell	male
Major League Soccer	male
Chris Laskowski	male
Io	female
Leo Maric	male

Names Without Gender
CNN
NASA
Gaming Humor
Electronic Arts
E3
Chicago Blackhawks
CAPTIVATING IMAGES
champayao
Transfer Daily
Adrenaline Rush
InfoSport Soccer
Momodu Jalloh
SportsCenter

Figure 7

Next, I redid all the relationships who was following who without the names that were null. Figure 8 shows part of the relationships and how they were organized to show that the source follows the target. I also created another csv file named Source_target_Q2 with the updated relationships as well.

source	target		
Ryan_Condotta	TheEPLVines	male	female
Ryan_Condotta	Trevora1William	male	male
Ryan_Condotta	NYMag	male	female
Ryan_Condotta	nytimes	male	female
Ryan_Condotta	easymoneymouss	male	male
Ryan_Condotta	ShaneCampbell23	male	male
Ryan_Condotta	MLS	male	male
Ryan_Condotta	CoachLaskowski	male	male
Ryan_Condotta	iokessler	male	female
Ryan_Condotta	leo_maric	male	male
Ryan_Condotta	s_perelman	male	male
Ryan_Condotta	MO_Jalloh09	male	male

Figure 8

In order to determine if the graph displays gender homophilly, I had to do several calculations. I first calculated the female to male, male to male, and female to female relationships and counted the total number of each of those relationships from the connections. Next, I calculated the total number male and female nodes for the graph. Lastly, I calculated p, q, 2pq, and Cross-Gender in order to determine gender homophilly. P is equal to the number of male nodes over the total. Q is equal to the number of female nodes over the total. 2pq is multiplying p and q times 2. Cross-Gender is the number of female to male relationships over the total number of relationships. Since Cross-Gender isn't significantly less than 2pq, my graph does not display gender homophilly. The tables of the calculations are shown in Figure 9.

Female to Male	Male to Male	Female to Female	Total
40	72	3	115

Male	Female	Total
26	11	37

p	q	2pq	Cross-Gender
0.702702703	0.297297297	0.417823229	0.347826087

Figure 9

Figure 10 shows the updated graph without the genders that were equal to null.

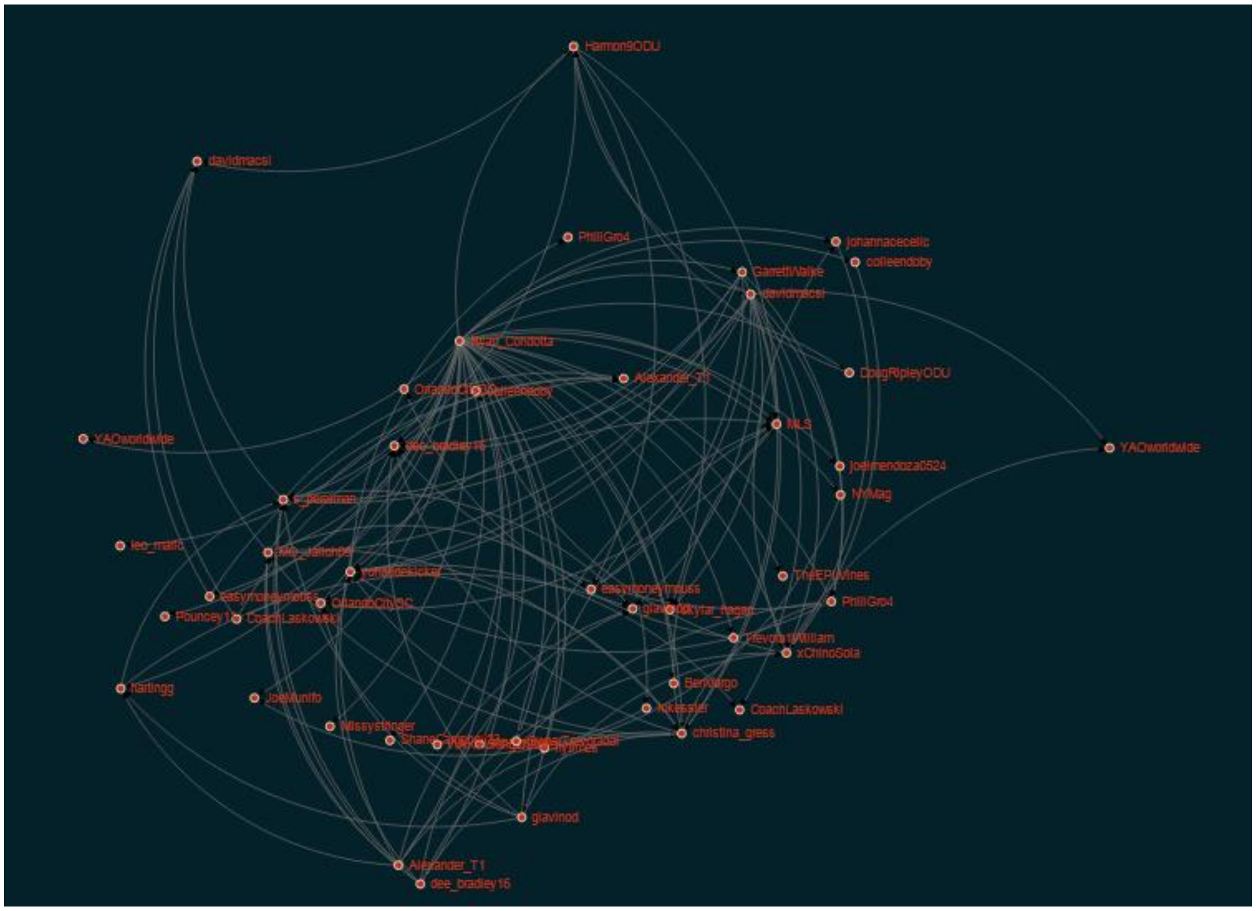

Figure 10