

CS-432/532 Introduction to Web Science:
Assignment #6:
Data Visualization

Due on Thursday, March 17, 2016

Dr. Michael L. Nelson

Plinio Vargas
pvargas@cs.odu.edu

Contents

Problem 1	1
1.1 Approach	1
1.1.1 Extracting the data	1
1.1.2 Plotting the Graph	4
1.2 Excluded Data	6
1.3 Solution	7
Problem 2	8
2.1 Approach	8
2.2 Excluded Data	9
2.3 Solution	10
2.3.1 Table of Twitter Users Likely Gender	10
2.3.2 Gender Graph URI	11
2.3.3 Test for Gender Homophily	11

List of Figures

1	Color Scheme	4
2	Jose A. Olvera Followers Network Graph	7
3	Gender Graph for Jose A. Olvera Twitter Followers	11

Listings

1	TwitterAPI.py	1
2	ReshapeData.py	4
3	GetGender.py	8
4	GenerateGender.py	9

List of Tables

1	Jose Olvera Twitter Followers Excluded Data	6
2	Gender Graph Excluded Data	9
3	Likely Gender for Jose Olvera's Twitter Followers	10

Problem 1

D3 graphing (5 points)

Use D3 to visualize your Twitter followers. Use my twitter account ("@phonedude_mln") if you do not have ≥ 50 followers. For example, @hvdsomp follows me, as does @martinkle1n. They also follow each other, so they would both have links to me and links to each other.

To see if two users follow each other, see: <https://dev.twitter.com/rest/reference/get/friendships/show>

Attractiveness of the graph counts! Nodes should be labeled (avatar images are even better), and edge types (follows, following) should be marked.

Note: for getting GitHub to serve HTML (and other media types), see: <http://stackoverflow.com/questions/6551446/can-i-run-html-files-directly-from-github-instead-of-just-viewing-their-source>

Be sure to include the URI(s) for your D3 graph in your report.

1.1 Approach

The twitter account for **Jose Antonio Olvera**, whom follows Dr. Nelson, was used as our data-set ***S*** for problem 1. This problem was divided into two sub-problems: extracting the data and plotting the graph.

1.1.1 Extracting the data

Listing 1 is a python program implemented to extract the data. It accomplishes the following:

- a. Obtains *S*'s followers Twitter's ID account using tweepy API [1]. Lines 10-16
- b. All IDs, including *S*, are placed into an array ***A***. Lines 20-21.
- c. Iterates through all elements in *A*, extracting its followers and other important information, such as name, screen name, number of friends, etc. Followers of each element are placed into an array ***B***. Lines 28-56
- d. Only *B*'s friends in *A* are needed, then $B = A \cup B$. Lines 36-39.
- e. The collection of all *Bs* arrays are saved into a JSON file. Lines 59-61.

Listing 1: TwitterAPI.py

```
10 auth = tweepy.OAuthHandler('w11PMTOUlj5sGIKjnSII7AcT9', '4
    if8m3WwRwvHFZ4wHKRXAgrb7mBuBRBoHKUDwoT4Afuol97hB')
11
12 api = tweepy.API(auth)
13
14 # screen_name = 'phonedude_mln'
15 screen_name = 'joc7188'
```

```

16 try:
17     user = api.get_user(screen_name)
18 except tweepy.TweepError as e:
19     if isinstance(e, tweepy.TweepError):
20         print('RateLimitError... Wait 15min')
21     sys.exit(1)
22
23 print(user.screen_name, user.id)
24 ids = [f for f in user.followers_ids()]
25
26 ids.append(user.id)      # add user_id to comparison list
27 user_friends = {}        # dictionary keeping all followers and user nodes
28 print(user.followers_count, ids)
29
30 counter = 0
31 excluded = []
32
33 for id_ in ids:
34     friend = api.get_user(id_)
35     print(counter, friend.name, friend.profile_image_url, friend.id, end=' ---> ')
36     counter += 1
37     user_connected_friends = []
38
39     try:
40         friend_followers = [f for f in api.get_user(friend.id).followers_ids()]
41         for connected in friend_followers:
42             if connected in ids:
43                 print(connected, end=',')
44                 user_connected_friends.append(connected)
45         print()
46         user_friends[friend.id] = {'name': friend.name, 'avatar': friend.
47                                     profile_image_url,
48                                     'screen_name': friend.screen_name, '
49                                     followers_count': friend.followers_count,
50                                     'friends_count': friend.friends_count, '
51                                     connected_to': user_connected_friends}
52
53     except tweepy.TweepError as e:
54         print('\n This is -->', e)
55         if isinstance(e, tweepy.TweepError):
56             time.sleep(60 * 15)
57
58         try:
59             # retry after 15 mins.
60             friend_followers = [f for f in api.get_user(friend.id).
61                                 followers_ids()]
62             for connected in friend_followers:
63                 if connected in ids:
64                     print(connected, end=',')
65                     user_connected_friends.append(connected)
66             print()
67             user_friends[friend.id] = {'name': friend.name, 'avatar': friend.
68                                         profile_image_url,

```

```
64         'screen_name': friend.screen_name, '
        'followers_count': friend.
        followers_count,
65         'friends_count': friend.friends_count,
        'connected_to':
        user_connected_friends}

66         continue
67
68     except tweepy.TweepError as e:
69         print('\n This is Other error in line 62-->', e)
70         excluded.append(friend.id)
71
72     else:
73         print('\n This is Other error in line 68 -->', e)
74         excluded.append(friend.id)
```

At the time our data-set has retrieved its size was of 53 elements. There is a request number limitations of 15, to extract Twitter’s user information. Lines 50-52 catches that error and wait for 15 minutes to continue with the information extraction.

Some Twitter users may have privacy settings in their accounts, so their information cannot be retrieved through the API. See section Excluded Data. Lines 68-74.

1.1.2 Plotting the Graph

Although we have the data from 1.1.1 to plot the graph, it does not have detail information on how nodes link. `< ReshapeData.py >` accomplishes the job. It creates within a file two JSON dictionaries: **nodes**, containing all elements characteristics and **links**, a source pointing to a target. It also adds a new characteristic to the node: color. It determines how close is the friendship between any node with **A**.

Figure 1: Color Scheme



Figure 1 represents a color scheme showing how similar is any particular node with data-set A in relation to its followers.

Listing 2: ReshapeData.py

```

46 """
47
48 Write to file
49 """
50 outputfile = 'jose.json'
51 with open(outputfile, 'w') as file:
52     file.write('{\n\t"nodes": [\n')
53     for record in data:
54         friendship = len(data[record]['connected_to']) / network_size
55         if friendship > 0.20:
56             friend_color = colors[0]
57         elif friendship > 0.14:
58             friend_color = colors[1]
59         elif friendship > 0.10:
60             friend_color = colors[2]
61         elif friendship > 0.04:
62             friend_color = colors[3]
63         else:
64             friend_color = colors[4]
65
66     print(friendship)
67     file.write('\t\t{\n\t\t\t"id": "%s",\n\t\t\t"name": "%s",\n\t\t\t"
68         followers_count": "%s",\'
69         '\n\t\t\t"friends_count": "%s",\n\t\t\t"screen_name": "%s",\n\t\t\t
70         '\t\t\t"avatar": "%s",\'
71         '\n\t\t\t"color": "%s"\n\t\t\t},\n' %

```

```

70         (record, data[record]['name'], data[record]['followers_count'], data
71           [record]['friends_count'],
72           data[record]['screen_name'], data[record]['avatar'], friend_color))
73     file.write('\t],\n\n\t"links": [\n')
74
75     for record in data:
76         for link in data[record]['connected_to']:
77             if str(link) in cross_idx:
78                 # file.write('\t\t{"source": %s, "target": %s, "type": "followed-by
79                 #   ", "value": %d},\n' %
80                   (cross_idx[str(record)], cross_idx[str(link)], 1))
81                 file.write('\t\t{"source": %s, "target": %s, "type": "followed-by
82                   ", "value": %d},\n' %
83                   (str(record), str(link), 1))
84     file.write('\t]\n')

```

In order to make the graph appealing in terms of friendship colors some test were conducted to capture different similarity levels. As a result, if any particular node has:

- If $> 20\%$ of N followers are also following A , then N is color coded as having similar followers as A .
- If node N has $> 14\%$ and $< 20\%$ of A followers then N is color coded as having most of its followers following A .
- If node N has $> 10\%$ and $< 14\%$ of A followers then N is color coded as having some its followers are following A .
- If node N 's has $> 4\%$ and $< 14\%$ of A followers then N is color coded as having few followers following A .
- If node N 's has $< 4\%$ of A followers then N is color coded as very few of N 's followers are following A .

The color code scheme is implemented in lines 49-62 of *ReshapeData.py*. The rest of the code just dump the data into *jose.json*, which is the input file for our Data-Driven Document. **The color code scheme is not accurate in a English grammar context, but it gives a quick view how closer are their followers network**

Data Visualization:

Various features from different D3JS sites were used to enhance graph appearance.

Zooming and dragging	http://bl.ocks.org/mbostock/6123708 [2]
Mouseover Tip-tool	http://bl.ocks.org/Caged/6476579 [3]
Directed graph	http://bl.ocks.org/mbostock/1153292 [4]
Mouseover link highlight	http://p.migdal.pl/wizualizacja-wolnych-lektur/polish_books_themes.html [5]
D3 Markers	http://bl.ocks.org/dustinlarimer/5888271 [6]

1.2 Excluded Data

Table 1: Jose Olvera Twitter Followers Excluded Data

ID	NAME	Screen-Name
509427317	Paulo Carrillo	panicape
271081991	monсорcas	monсорcas

Values shown in Table 1 are followers of Jose Olvera Twitter's Account, but were not included because these accounts have a private settings and cannot be extracted without authentication.

1.3 Solution

Figure 2: Jose A. Olvera Followers Network Graph

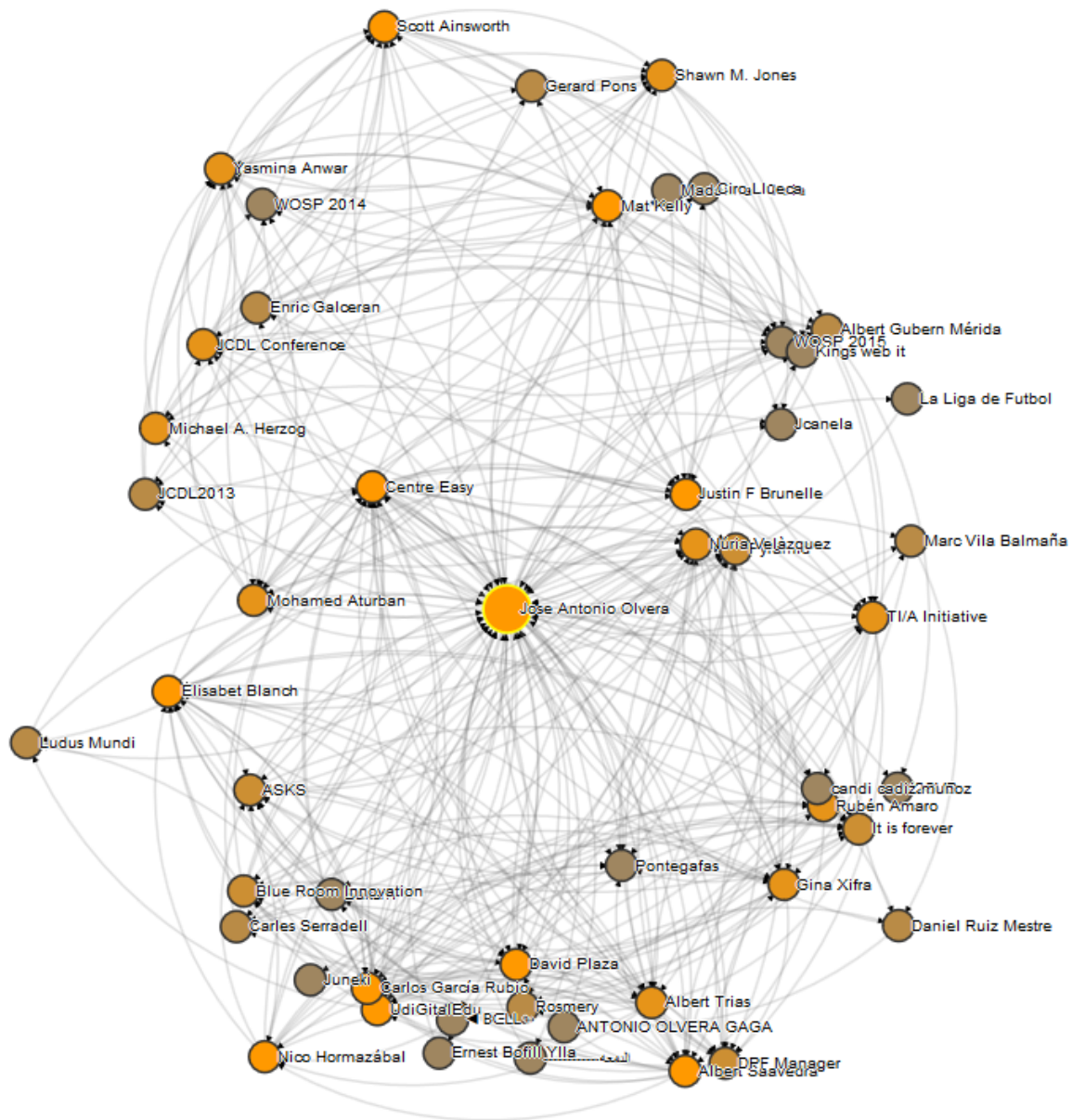


Figure 2 represents a color scheme showing how similar are the followers of any particular node with A followers. Node A (Jose Antonio Olvera) has a larger radius and a yellow stripe around to distinguish it from the rest of the nodes in the graph.

Problem 2

Gender homophily in your Twitter graph (5 points)

Take the Twitter graph you generated in question #1 and test for male-female homophily. For the purposes of this question you can consider the graph as undirected (i.e., no distinction between "follows" and "following"). Use the twitter name (not "screen name"; for example "Michael L. Nelson" and not "@phone-dud_mln") and programatically determine if the user is male or female. Some sites that might be useful:

<https://genderize.io/>
<https://pypi.python.org/pypi/gender-detector/0.0.4>

Create a table of Twitter users and their likely gender. List any accounts that can't be determined and remove them from the graph.

Perform the homophily test as described in slides 11-15, Week 7.

Does your Twitter graph exhibit gender homophily?

2.1 Approach

We we used the same data generated in problem 1 to solve this problem. However, we filtered the data set by injecting the first name of each node with gender_detector python library. The iteration is shown in the listing below:

Listing 3: GetGender.py

```
20 detector = GenderDetector('us')
21
22 with open("jose.json", 'rb') as file:
23     data = json.load(file)
24
25 counter = 0
26 unknown = []
27 with open("jose-gender.data", 'w') as file:
28     for name in data['nodes']:
29         counter += 1
30         try:
31             print(counter, name['name'].split()[0].encode('ascii'), detector.guess
32                   (name['name'].split()[0]), name['id'])
33             file.write('%s, %s, %s\n' % (str(name['id']), name['name'].split()[0].
34                                         encode('ascii'), detector.guess(name['name'].split()[0])))
35         except:
36             print(counter, name['name'].split()[0], 'unknown')
37             file.write('%s, %s, unknown\n' % (str(name['id']), name['name'].split
38                                               ([0].encode('utf-8'))))
```

The result file can be seen on Table 3. To complete our graph and generate a solution we use the same approach as in 1.1.2, but we added an extra field to the nodes: gender, and we placed all linking edges into a set, thus eliminating the extra edge between two nodes having a bi-directional relationship. Since the JSON generation file approach is the same as in previous problem, only where they differ will be pointed out.

Listing 4: GenerateGender.py

```

52 counter = 0
53 linked_nodes = set()
54 for node in data['links']:
55     if str(node['target']) in gender_nodes and str(node['source']) in gender_nodes
56         :
57         if (node['source'], node['target']) not in linked_nodes and (node['target',
58             ], node['source']) not in linked_nodes:
59             counter += 1
60             print('\t\t{"source": %s, "target": %s, "type": "followed-by"},\n' %
61                 (node['source'], node['target']), end='')
62             linked_nodes.add((node['source'], node['target']))
63 print('\t\n', end='')

```

As mentioned before, a set object was created in line 53 and we check in both sides of linked nodes (lines 55-56) to verify if the relationship exists.

Data Visualization:

We used same approach as in 1.1.2, difference our color scheme was base on two colors: male (blue) and female (pink).

2.2 Excluded Data

Table 2: Gender Graph Excluded Data

ID	First-Name	Gender	ID	First-Name	Gender
467244543	It	unknown	1364325193	Juneki	unknown
22826489	Shawn	unknown	861118038	Jcanela	unknown
1919132468	Ludus	unknown	3271976762	Kings	unknown
3094776599	WOSP	unknown	256914642	Pyramid	unknown
20786017	Nico	unknown	2341223632	unicode	unknown
4255657283	Blue	unknown	299702944	UdiGitalEdu	unknown
204062965	Rubén	unknown	2585451218	I2CVB	unknown
320655858	Centre	unknown	4070742673	unicode	unknown
2986305078	JCDL	unknown	2512551954	WOSP	unknown
49576758	Enric	unknown	3200493898	DPF	unknown
606550203	JCDL2013	unknown	295664883	Núria	unknown
3200537140	TI/A	unknown	898245566	Pontegafas	unknown
506173851	ASKS	unknown	2220440436	unicode	unknown
80852622	La	unknown			

Gender for data-set above were not able to be recognized by the python library gender_detector

2.3 Solution

2.3.1 Table of Twitter Users Likely Gender

Table 3: Likely Gender for Jose Olvera's Twitter Followers

ID	First-Name	Gender	ID	First-Name	Gender
120440596	David	male	857229727	ANTONIO	male
467244543	It	unknown	180473295	Justin	male
3787923975	candi	female	48938481	Albert	male
22826489	Shawn	unknown	506173851	ASKS	unknown
387831845	Michael	male	80852622	La	unknown
117034312	Ciro	male	862861328	Daniel	male
1919132468	Ludus	unknown	1364325193	Juneki	unknown
1511472330	Mohamed	male	861118038	Jcanela	unknown
479894950	Mat	male	3271976762	Kings	unknown
3094776599	WOSP	unknown	256914642	Pyramid	unknown
397587860	Rosmery	female	2341223632	unicode	unknown
199226328	Marc	male	299702944	UdiGitalEdu	unknown
373298265	Gina	female	3082680478	Carles	male
551758619	Carlos	male	2585451218	I2CVB	unknown
367167714	Ernest	male	4070742673	unicode	unknown
20786017	Nico	unknown	2512551954	WOSP	unknown
551038996	Albert	male	3200493898	DPF	unknown
4255657283	Blue	unknown	282698602	Elisabet	female
204062965	Rubén	unknown	48842695	Gerard	male
320655858	Centre	unknown	295664883	Núria	unknown
2986305078	JCDL	unknown	114584669	Albert	male
49576758	Enric	unknown	898245566	Pontegafas	unknown
365952591	Jose	male	11938602	Madalina	female
606550203	JCDL2013	unknown	2220440436	unicode	unknown
32747579	Scott	male	221460308	Yasmina	female
3200537140	TI/A	unknown			

Most likely gender was generated utilizing python library gender_detector

2.3.2 Gender Graph URI

Figure 3: Gender Graph for Jose A. Olvera Twitter Followers

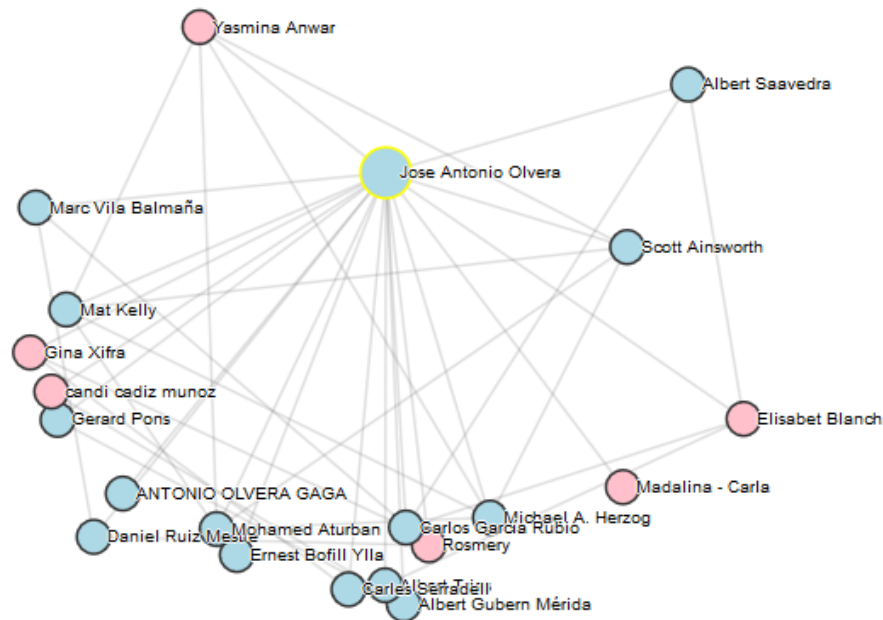


Figure 3 visualize gender relationship among Jose Olvera Followers.

2.3.3 Test for Gender Homophily

To test our graph for gender homophily we implemented `< homophily.py >`. The result of running the program is shown below:

```
Starting Time: Wed, Mar 16, 2016 at 21:50:36
```

```
Number of females: 6
```

```
Number of males: 18
```

```
p: 0.750
```

```
q: 0.250
```

```
2pq: 0.375
```

```
Cross-gender edges:
```

```
-----
```

```
Elisabet Blanch <--> David Plaza
```

```
Yasmina Anwar <--> Michael A. Herzog
```

```
Yasmina Anwar <--> Mohamed Aturban
Yasmina Anwar <--> Mat Kelly
Jose Antonio Olvera <--> Rosmery
Daniel Ruiz Mestre <--> Rosmery
Marc Vila Balmana <--> Rosmery
Carlos Garcia Rubio <--> Gina Xifra
Jose Antonio Olvera <--> Gina Xifra
Carles Serradell <--> Gina Xifra
Elisabet Blanch <--> Carlos Garcia Rubio
Elisabet Blanch <--> Albert Trias
candi cadiz munoz <--> Jose Antonio Olvera
Yasmina Anwar <--> Jose Antonio Olvera
Madalina - Carla <--> Jose Antonio Olvera
Elisabet Blanch <--> Jose Antonio Olvera
Yasmina Anwar <--> Scott Ainsworth
Yasmina Anwar <--> Justin F Brunelle
candi cadiz munoz <--> Albert Gubern Merida
Albert Saavedra <--> Elisabet Blanch
```

```
Summary of Cross-gender edges: 20 out of 52
Percentage of Cross-gender edges 0.385
```

```
End Time: Wed, Mar 16, 2016 at 21:50:36
Execution Time: 0.00 seconds
```

As we can see $2pq < \text{cross-edges}$, $0.375 < 0.385$ \therefore there is no evidence of homophily.

References

- [1] Tweepy API. (n.d.) Retrieved March 10, 2016, from <http://docs.tweepy.org/en/latest/api.html>
- [2] Zooming and dragging. (n.d.) Retrieved March 10, 2016, from <http://bl.ocks.org/mbostock/6123708>
- [3] d3-tip tool. (n.d.) Retrieved March 10, 2016, from <http://bl.ocks.org/Caged/6476579>
- [4] Directed graph. (n.d.) Retrieved March 10, 2016, from <http://bl.ocks.org/mbostock/1153292>
- [5] Polish Books. (n.d.) Retrieved March 10, 2016, from http://p.migdal.pl/wizualizacja-wolnych-lektur/polish_books_themes.html
- [6] D3 Markers. (n.d.) Retrieved March 10, 2016, from <http://bl.ocks.org/dustinlarimer/5888271>