

CS532 Web Science: Assignment 6

Finished on March 17, 2016

Dr. Michael L. Nelson

Naina Sai Tipparti
ntippart@cs.odu.edu

Contents

Problem 1	2
Question	2
Answer	2
Problem 2	6
Question	6
Answer	6
Problem 3	10
Question	10
Answer	10

Listings

1	Get followers of followers	2
2	Who follows who graph file label	4
3	gender.py	6
4	homophily.py	9
5	Data Converter	12
6	Building the Graph	13

List of Figures

1	Naina Sai Tipparti Followers Network Graph	5
2	Initial Graph	10
3	Graph After Split	11

Problem 1

Question

Use D3 to visualize your Twitter followers. Use my twitter account (“@phonedude_mln”) if you do not have ≥ 50 followers. For example, @hvdsonp follows me, as does @mart1nkle1n. 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.

Answer

With the use of D3.js (a JavaScript library for manipulating documents based on data) [?], and the Force-Directed Graph template for directed graphs (<http://bl.ocks.org/mbostock/1153292>), following solution was achieved.

The solution for this problem is outlined by the following steps:

1. **Develop an python script to get the followers of followers:** In order to get the followers of followers across a variable degree, I derived the iterative python script as shown in Listing 1

```

1  import time
   import tweepy
   import json
   import sys
   import os
6  import re

   consumer_key = "mvW9Y4uy8xJWMNwqV97qCa2aX"
   consumer_secret = "vyCabSoD6CPXeAdL7onyEGO6lBl6YyPeivLpng1MgM2bOBjisU"
   access_token = "798668178-bH8DbMpNuWkfhAHxuODgWSHwQE65B1WZnc4Ahtej"
11  access_token_secret = "FhykPKnQcgKQBE43os2bDZ31ugH9RVSG3HYoOL7QG7RNC"

   auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
   auth.set_access_token(access_token, access_token_secret)

16  api = tweepy.API(auth)
   screen_name='9ulovesu'

   try:
       user = api.get_user(screen_name)
21  except tweepy.TweepError as e:
```

```

        if isinstance(e, tweepy.TweepError):
            print('RateLimitError')

        sys.exit(1)
26
    print(user.screen_name, user.id)
    ids = [f for f in user.followers_ids()]

    ids.append(user.id)
31
    user_friends = {}
    print(user.followers_count, ids)

    counter = 0
    excluded = []
36
    for id_ in ids:
        friend = api.get_user(id_)
        print(counter, friend.name, friend.profile_image_url, friend.id)
        print('—>')
41
        counter += 1
        user_connected_friends = []

        try:
            friend_followers = [f for f in api.get_user(friend.id).followers_ids()]
46
            for connected in friend_followers:
                if connected in ids:
                    print(connected)
                    print(",")
                    user_connected_friends.append(connected)
51
            user_friends[friend.id] = {'name': friend.name, 'avatar': friend.
                profile_image_url, 'screen_name': friend.screen_name, 'followers_count':
                friend.followers_count, 'friends_count': friend.friends_count, '
                connected_to': user_connected_friends}
            print(user_friends[friend.id])
        except tweepy.TweepError as e:
            print('\n This is —', e)
            if isinstance(e, tweepy.TweepError):
56
                time.sleep(60 * 15)
                try:
                    friend_followers = [f for f in api.get_user(friend.id).followers_ids()]
                    ]
                    for connected in friend_followers:
                        if connected in ids:
61
                            print(connected)
                            print(",")
                            user_connected_friends.append(connected)
                    user_friends[friend.id] = {'name': friend.name, 'avatar': friend.
                        profile_image_url, 'screen_name': friend.screen_name, '
                        followers_count': friend.followers_count, 'friends_count': friend.
                        friends_count, 'connected_to': user_connected_friends}
                    print(user_friends[friend.id])
66
                    continue

                except tweepy.TweepError as e:
                    print(e)
                    excluded.append(friend.id)
71
            else:
                print(e)
                excluded.append(friend.id)

    with open("9ulovesu.json", 'w') as file:
76
        data = json.load(file)

```

Listing 1: Get followers of followers

2. **Visualize graph:** With the use of the graph template at <http://bl.ocks.org/>

mbostock/1153292, I replaced the default data in the JSON file with the output of followers as outlined in Listing ??.

3. Excluded Data Values shown in Table 1 are followers of Naina Sai Tipparti Twitter's

ID	NAME	Screen-Name
152684352	Anudeep	anuhearthacker
2538627170	ÃÃmitha SophiaD'Souza	DSouzMitha
2433403314	GK Tecvision	GKTecvision

Table 1: Naina Sai Tipparti Twitter Followers Excluded Data

Account, but were not included because these accounts have a private settings and cannot be extracted without authentication.

```

4      "id": 2755649263,
      "screen_name": "RithikaR9",
      "friends_count": 58,
      "connected_to": [798668178, 54493821, 118623489, 157985123],
      "followers_count": 5,
      "avatar": "http://abs.twimg.com/sticky/default_profile_images/
      default_profile_3_normal.png",
      "name": "Rithika Reddy"
9    }, {
      "id": 54493821,
      "screen_name": "Manoj_Chandral1",
      "friends_count": 157,
      "connected_to": [2755649263, 1875320502, 147860404, 335970153, 798668178,
14      709467001421500416, 118623489, 157985123],
      "followers_count": 54,
      "avatar": "http://pbs.twimg.com/profile_images/709387424477155328/
      HTm2wPZN_normal.jpg",
      "name": "manoj Kompalli"
19    }, {
      "id": 118623489,
      "screen_name": "dineshpaladhi",
      "friends_count": 207,
      "connected_to": [798668178, 709467001421500416, 54493821, 2755649263,
24      157985123],
      "followers_count": 61,
      "avatar": "http://abs.twimg.com/sticky/default_profile_images/
      default_profile_5_normal.png",
      "name": "dinesh kumar paladhi"
29    }, {
      "id": 335970153,
      "screen_name": "abhipolavarapu",
      "friends_count": 21,
      "connected_to": [798668178, 54493821],
      "followers_count": 11,
      "avatar": "http://pbs.twimg.com/profile_images/532157901696028672/
      XltjrHmY_normal.jpeg",
      "name": "Abhishek Polavarapu"
    }, {
      "id": 709467001421500416,

```

Listing 2: Who follows who graph file label

The result of running followers.py as in Listing 1 and figure 1 can be seen at <http://www.cs.odu.edu/~ntippart/cs532/a6/followers.html>

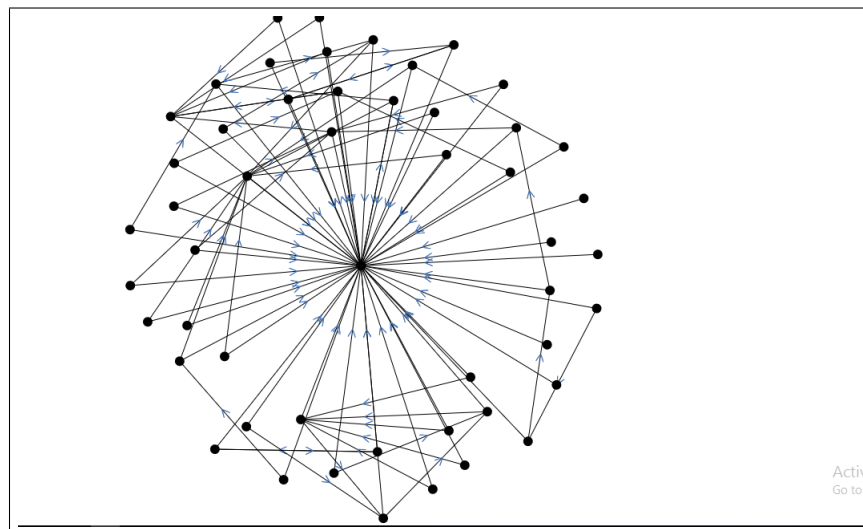


Figure 1: Naina Sai Tipparti Followers Network Graph

Problem 2

Question

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 “@phonedude_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?

Answer

Develop an python script to get the gender: We we used the same data generated in problem 1 to solve this problem. The iteration is shown in the listing 3. However, we filtered the data set by injecting the first name of each node with gender_detector python library.

```
import os
import sys
import json
from gender_detector import GenderDetector
5 from time import strftime, localtime, time

# record running time
start = time()
print('Starting Time: %s' % strftime("%a, %b %d, %Y at %H:%M:%S", localtime()))
10 detector = GenderDetector('us')

with open("9ulovesu.json", 'rb') as file:
    data = json.load(file)

15 counter = 0
unknown = []
with open("9ulovesu.data", 'w') as file:
    for name in data['nodes']:
        counter += 1
20     try:
        print(counter, name['name'].split()[0].encode('ascii'), detector.guess(name['name'].split()[0], name['id']))
        file.write('%s, %s, %s\n' % (str(name['id']), name['name'].split()[0].encode('ascii'), detector.guess(name['name'].split()[0])))
    except:
        print(counter, name['name'].split()[0], 'unknown')
25     file.write('%s, %s, unknown\n' % (str(name['id']), name['name'].split()[0].encode('ascii-8')))
```

```
print('\nEnd Time: %s' % strftime("%a, %b %d, %Y at %H:%M:%S", localtime()))
print('Execution Time: %.2f seconds' % (time()-start))
```

Listing 3: gender.py

Table 2: Naina Sai Tipparti Twitter Followers

ID	Name	Gender
3185979164	Ganesh	male
1877482201	Shonna	female
2822485029	neta	female
3148781365	sophia	female
3115151598	adityajavvaji3	unknown
1498767308	kOUSHIk	male
3106046126	SAMA	female
3039676646	S	unknown
3023712804	Vishruth	male
108917717	shreesh	unknown
1475712962	vishal	male
2940668600	tejaswi.tipparti@gma	unknown
32536673	David	male
2920714134	Avinash	male
896783850	shashank	male
2895776695	Abhradipta	unknown
2855422626	vivekreddy	unknown
123828688	shashank	male
87169236	karthik	male
755566003	Bakakrishna	unknown
469972014	Rev@nth	unknown
822761454	Nilesh	male
429273675	Aditeswar	unknown
2583173624	Jagan	male
2575092290	kona	unknown
2451909986	charan	male
2439592034	venkat	male
2243833796	Ratna	unknown
111850945	lalit	male
335158405	charan	male

ID	Name	Gender
1128679802	Aasresh	unknown
1071911132	Arun	male
798668178	Naina	female
2298403772	WinBuzzer	unknown
3406851292	bharath	male
1407026076	Keyu	unknown
2755649263	Rithika	female
54493821	manoj	male
118623489	dinesh	male
335970153	Abhishek	male
709467001421500416	Shivani	female
1875320502	sai	unknown
157985123	majetisiri	unknown
4792335737	Growth	unknown
23829268	Markus	male
2811553591	Anvesh	unknown
108313439	Krishna	unknown
2665053835	rio	unknown
470149644	Cathryn	female
147860404	Sumanth	male
4169240239	iamsonofgodjesus	unknown
4026956354	kannan	male
3746984594	Mattada.Rajesh	unknown
130118907	Vijay	male
3303591120	durgam	unknown
3277164888	Vinodh	unknown
3276929190	Siri	female
3238430006	Harish	male
3181275751	Aasresh	unknown
3223939152	sandy	unknown
1588702699	NuDawn	unknown
3189658530	nikhil	male

```
import os
2 import sys
import json

gender = {'female': 0, 'male': 0}
node_gender = {}
7 number_cross_gender = 0

with open("9ulovesu-gender.json", 'r') as file:
    data = json.load(file)

12 for node in data['nodes']:
    node_gender[node['id']] = {'gender': node['gender'], 'name': node['name']}
    if node['gender'] == 'female':
        gender['female'] += 1
    elif node['gender'] == 'male':
17         gender['male'] += 1

n = gender['male'] + gender['female']
p = gender['male'] / n
q = gender['female'] / n
22

print('\nNumber of females: %d' % gender['female'])
print('Number of males: %d' % gender['male'])
print('p: %.3f' % p)
print('q: %.3f' % q)
27 print('2pq: %.3f' % (2 * p * q))
```

Listing 4: homophily.py

```
Number of females: 9
Number of males: 26
p: 0.742
q: 0.257
2pq: 0.381
```

Problem 3

Question

Using D3, create a graph of the Karate club before and after the split.

- Weight the edges with the data from:

<http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/zachary.dat>

- Have the transition from before/after the split occur on a mouse click. This is a toggle, so the graph will go back and forth between connected and disconnected.

Answer

Using an example from the primary author of the D3 JavaScript library, Mike Bostok [?], a graph was created of Zachary's Karate Club using the pickled [?] dataset found at http://nexus.igraph.org/api/dataset_info?id=1&format=html. The D3 library provides a force-directed graphing layout [?], which was used to display the graph. A transition from the initial graph, shown in Figure 2, to the graph after the split of the karate club, shown in Figure 3, was created using standard JavaScript.

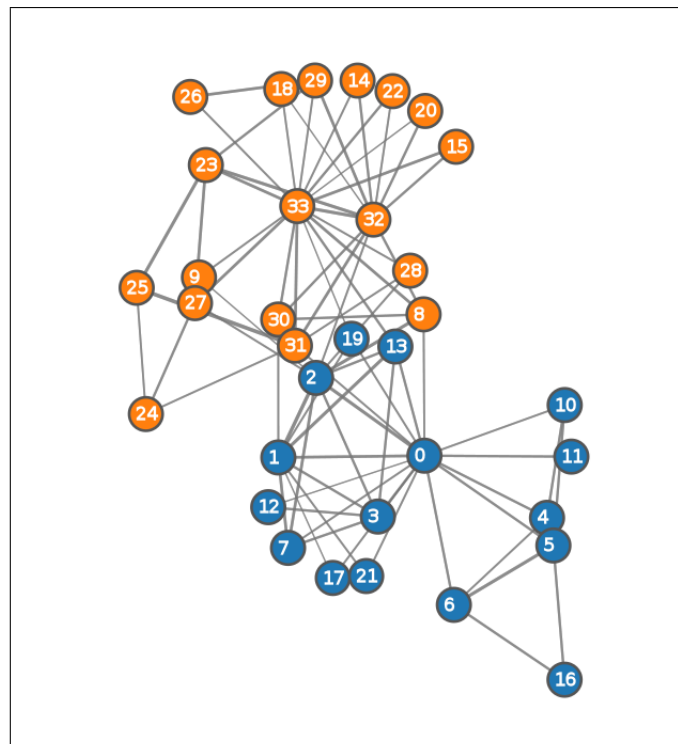


Figure 2: Initial Graph

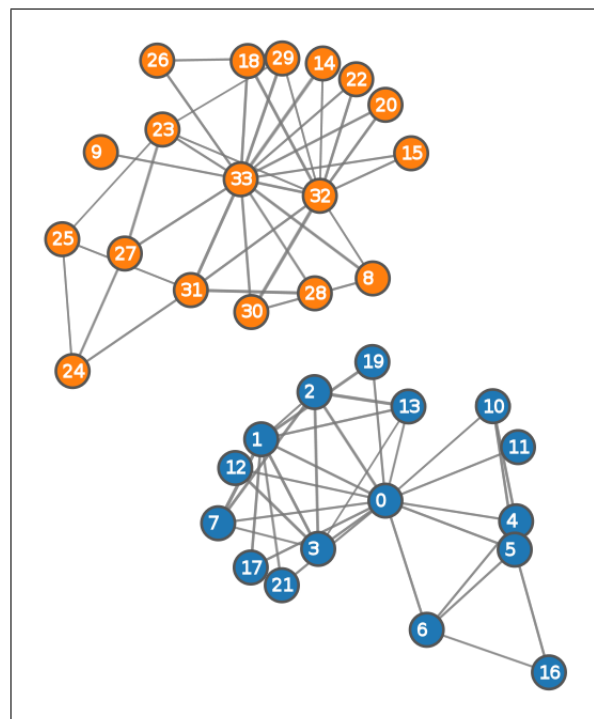


Figure 3: Graph After Split

The dataset was first parsed into matrices using the `build_matrix` function, shown in Listing 5. These matrices were converted into python dictionary objects, which are pickled [?] into the json format [?]. This output was used as the input for the JavaScript code, which uses the D3 library [?] to create the graphs.

The python code to produce the json data is shown in Listing 5.

```

5  #!/usr/bin/env python

import pickle
import json

def build_matrix(raw, n):
    """Builds a matrix from a list of strings where each string
    is a space-separated row of the matrix"""
    matrix = [[0 for y in range(n)] for x in range(n)]
10     for idx, line in enumerate(raw):
        for idy, val in enumerate(line.split()):
            matrix[idx][idy] = int(val)
    return matrix

15 def build_nodes(clubs, names):
    """Builds a list of nodes represented as a python dict"""
    return [{"id": i, "club": c, "name": names[i]} for i, c in enumerate(clubs)]

def build_links(egraph, cgraph):
    """Builds a list of links represented as a python dict"""
20     links = []
    for idx, line in enumerate(egraph):
        for idy, val in enumerate(line):
            if val == 1:
25                 links.append({"source": idx, "target": idy, "value": cgraph[idx][idy]})
    return links

if __name__ == '__main__':
    data = pickle.loads(open('karate.pickle').read())['karate']
30     clubs = data.vs['Faction']
    names = data.vs['name']

    # Read lines from input data file
    lines = [line.strip() for line in open('karate.txt').readlines()]
35     # parse size of nxn matrix
    n = int(lines[1].split()[0].replace('N=', ''))

    # build matrices using input data
40     egraph = build_matrix(lines[7:41], n)
    cgraph = build_matrix(lines[41:], n)

    m = {}
    m['nodes'] = build_nodes(clubs, names)
45     m['links'] = build_links(egraph, cgraph)
    with open('out.json', 'w') as output:
        output.write(json.dumps(m, indent=1, separators=(',', ': ')))

```

Listing 5: Data Converter

The JavaScript code to produce the graph is shown in Listing 6.

```

var width = 960,
    height = 700;

30 var color = d3.scale.category10();

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

35 var force = d3.layout.force()
    .charge(-400)
    .gravity(0.1)
    .linkDistance(90)
    .size([width, height]);

40 d3.json("out.json", function(error, graph) {
    var link = svg.selectAll(".link")
        .data(graph.links)
        .enter().append("line")
        .attr("class", "link")
        .style("stroke-width", function(d) { return Math.sqrt(d.value); });

    var node = svg.selectAll(".node")
        .data(graph.nodes)
        .enter().append("g")
        .attr("class", "node")
        .call(force.drag);

    55 node.append("circle")
        .attr("r", 12)
        .style("fill", function(d) { return color(d.club); });

    node.append("text")
        .attr("dx", -7)
        .attr("dy", ".35em")
        .text(function(d) { return d.id; });

    function update() {
    65     node.data(graph.nodes)
        .exit().remove();
        link.data(graph.links)
        .exit().remove();
        force.nodes(graph.nodes)
        .links(graph.links)
        .on("tick", tick)
        .start();
    };

    75 function tick() {
        link.attr("x1", function(d) { return d.source.x; })
            .attr("y1", function(d) { return d.source.y; })
            .attr("x2", function(d) { return d.target.x; })
            .attr("y2", function(d) { return d.target.y; });

        node.attr("transform", function(d) {
            80             return "translate(" + d.x + "," + d.y + ")";
        });
    };

    85 var done = false;
    svg.on("click", function(d) {
        if(!done) {
            done = true;
            90 graph.links = graph.links.filter(function(d) {
                return graph.nodes[d.source.id].club == graph.nodes[d.target.id].club;
            });
        }
    });

```

```
95      update();  
      }  
    });  
    update();  
  });
```

Listing 6: Building the Graph

