Assignment 6
CS532, Web Science, Spring 2017
Old Dominion University, Computer Science Dept

Hussam Hallak
CS Master's Student
Prof: Dr. Nelson

Question 1:

Use D3 to visualize your Twitter followers. Use my twitter account ("@phonedude_mln") if you do not have 50 followers or more. For example, @hvdsomp 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:

I do not have a minimum of 50 followers on Twitter, however I have an old account and accumulated 42 followers, which is close enough, and it will also help me in the next step because it is taking days to extract data using Twitter API if I used "phonedude_mln". I created a python script, getfollowers.py, that will get all followers of a Twitter user passed as a command line argument; in this case it is "iHussam". The program saves all usernames of the followers in a text file. The first username in the file is the user passed in the command line, "iHussam". The output file name has the format ¡username¿FOLLOWERS.txt. In this case it is "iHussamFOLLOWERS.txt".

Listing 1: The content of getfollowers.py

```
import sys
import time
import tweepy

if len(sys.argv) < 2:
   print "Usage: Python getfollowers.py <twitter_userid>"
   print "e.g: Python getfollowers.py phonedude_mln"
   exit()
username = sys.argv[1]
#config.py contains your twitter's API consumer_key, consumer_secret, access_key,
     and access_secret
config = {}
execfile("config.py", config)
auth = tweepy.OAuthHandler(config["consumer_key"], config["consumer_secret"])
auth.set_access_token(config["access_key"], config["access_secret"])
api = tweepy.API(auth)

users = tweepy.Cursor(api.followers, screen_name=username).items()

filename = username + 'FOLLOWERS.txt'
```

```
fh_output = open(filename, "w")
fh_output.write(username + '\n')

while (1):
    try:
        user = next(users)
   fh_output.write(user.screen_name + '\n')
    except tweepy.TweepError:
        time.sleep(60)
fh_output.close()
```

Listing 2: Running getfollowers.py

```
root@ima-app:/var/www/Hussam/A6# python getfollowers.py iHussam
root@ima-app:/var/www/Hussam/A6# cat iHussamFOLLOWERS.txt
iHussam
HussamHallak1
mohamed29259566
miloudkhammame
alloshasslan
shahzad80803670
buffalcabreured
atifsheikh78663
husseinachabani
mouaz_2012
mohzhran
so_love_kevin
didepolool
IkramDreamer
TizimiPress
adelmg8
Jenny867530911
Hassanalhellaly
mmb_badenjki
AbdallaMoody
REESEDASHOOTA
FreelancingSkul
amino_chinwi_
AhnoussHicham
CSSLight
stevevanleeuwen
power008
abu_aleppo
YassineMazal
langkawi2030
funcamrocks
Tidewaterns
9_willy
Smil1m
frooja2011
TheCoder1
majebry
mohawow
AbdelazizOuaiji
ro2yahcom
```

```
tarekfouad3
faisalhakem
EvaGregory
```

Next step is to find out who follows who among my followers. I wrote a python script that using Twitter API. The script file is named "whofollowswho.py". It takes a text file that has all usernames we want to find the friendship among them as command line parameter, and it generates two comma separated .csv files as output, one of which contains the nodes "nodes.csv", and the other contains the edges "links.csv". These two file will be the input files for "index.html" that will have the graph in D3.

Listing 3: The content of whofollowswho.py

```
import sys
import time
from twitter import *
import twitter

if len(sys.argv) < 2:
   print "Usage: python whofollowswho.py <file_name>"
   print "e.g: python whofollowswho.py phonedude_mlnFOLLOWERS.txt"
   exit()

filename = sys.argv[1]
fh_input = open(filename, 'r')
links_output = open("links.csv", 'w')
links_output.write('source,target,type\n')
nodes_output = open("nodes.csv", 'w')
nodes_output.write('node,group\n')
users = fh_input.readlines()
#config.py contains your twitter's API consumer_key, consumer_secret, access_key,
     and access_secret
config = {}
execfile("config.py", config)
api = twitter.Api(consumer_key=config["consumer_key"], consumer_secret=config["
    consumer_secret"], access_token_key=config["access_key"], access_token_secret
    =config["access_secret"], sleep_on_rate_limit=True)
for user in users:
   user = user.strip()
   nodes_output.write(str(user) + ',group1\n')
for user_A in users:
   user_A = user_A.strip()
   for user_B in users:
      user_B = user_B.strip()
      try:
                result = api.ShowFriendship(source_screen_name = user_A ,
                    target_screen_name = user_B)
                following = result["relationship"]["target"]["following"]
                if following:
           links_output.write(user_A + ',' + user_B + ',type1\n')
              except Exception as err:
                print(err)
fh_input.close()
links_output.close()
nodes_output.close()
```

Listing 4: Running whofollowswho.py

```
root@ima-app:/var/www/Hussam/A6# python whofollowswho.py iHussamFOLLOWERS.txt
```

After running the script "whofollowswho.py", we have all the data needed to generate the graph in D3. The code is in the file "index.html".

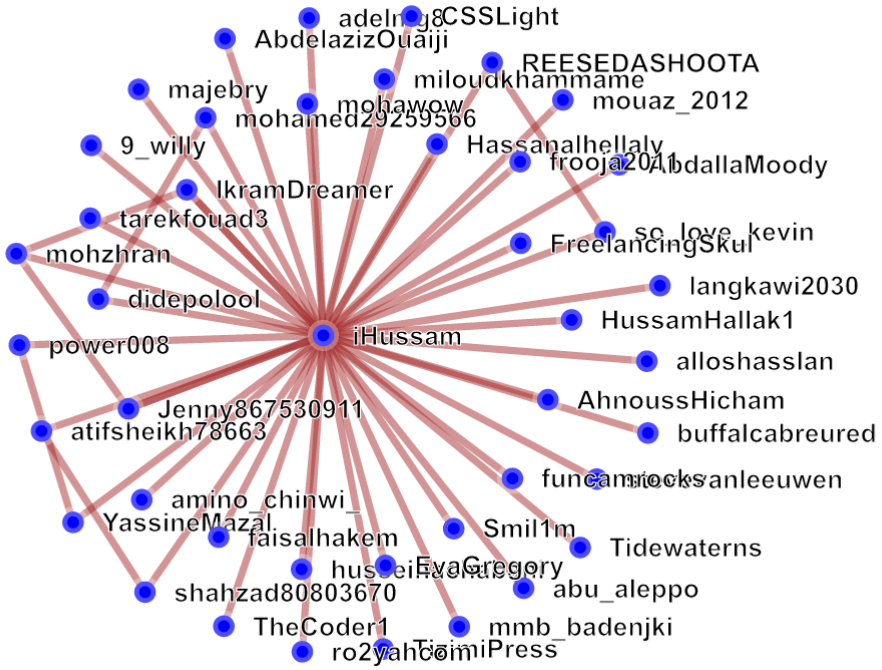
Listing 5: The content of index.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>
.link {
  stroke: #999;
  stroke-opacity: .6;
}
.node {
  stroke: #fff;
  stroke-width: .5px;
}
.node text {
  pointer-events: none;
  font: 11px sans-serif;
  font-weight: bolder;
}
.node.group1{
  fill: blue;
  stroke-opacity: .3;
  stroke-width: 4;
}
.link.type1{
  stroke: brown;
  stroke-opacity: .5;
  stroke-width: 3;
}
</style>
<body>
<script type="text/javascript" src="http://d3js.org/d3.v3.min.js"></script>
<script>

var linkpath = ("links.csv");
var nodepath = ("nodes.csv");

var width = 1200,
    height = 700;

var color = d3.scale.category20();

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

//Want to have different labels
// SETTING UP THE FORCE LAYOUT
  var force = d3.layout.force()
  //using width/height from above, but size is mainly det'd by linkDistance and
```

```
      charge
    .size([width, height])
    // how far between nodes
    .linkDistance(80)
    // changes how close nodes will get to each other. Neg is farther apart.
    .charge(-300);

d3.csv(nodepath, function(nodes) {

  var nodelookup = {};
  var nodecollector = {};

   count = 0;
// we want to create a lookup table that will relate the links file and the nodes
     file
    nodes.forEach(function(row) {
    nodelookup[row.node] = count;

    nodecollector[row.node] = {name: row.node, group: row.group};
    //console.log(nodecollector)

    //console.log(row.node)
    //console.log(nodelookup)

    count++;
 });

//Get all the links out of of the csv in a way that will match up with the nodes

d3.csv(linkpath, function(linkchecker) {

  var linkcollector = {};
  indexsource = 0;
  indextarget = 0;
    count= 0;
    //console.log(nodelookup['celery'])
    linkchecker.forEach(function(link) {

   linkcollector[count] = {source: nodelookup[link.source], target: nodelookup[
       link.target], type: link.type };
    //console.log(linkcollector[count])
   count++
});

//console.log(linkcollector)
var nodes = d3.values(nodecollector);
var links = d3.values(linkcollector);

//console.log(nodes)
//console.log(links)

// arrow business
// build the arrow.
svg.append("svg:defs").selectAll("marker")
    .data(["end"])
  .enter().append("svg:marker")
```

```
    .attr("id", String)
    .attr("viewBox", "0 -5 10 10")
    .attr("refX", 15)
    .attr("refY", -1.5)
    .attr("markerWidth", 6)
    .attr("markerHeight", 6)
    .attr("orient", "auto")
  .append("svg:path")
    .attr("d", "M0,-5L10,0L0,5");

// add the links and the arrows
var path = svg.append("svg:g").selectAll("path")
    .data(force.links())
  .enter().append("svg:path")
    .attr("class", "link")
    .attr("marker-end", "url(#end)");

//end arrow business

  // Create the link lines.
  var link = svg.selectAll(".link")
      .data(links)
     .enter().append("line")
     .attr("class", function(d) { return "link " + d.type; })

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

 //put in little circles to drag
  node.append("circle")
      .attr("r", 4.5)
    .attr("class", function(d) { return "node " + d.group; })
    .call(force.drag);

//add the words
 node.append("text")
      .attr("dx", 12)
      .attr("dy", ".35em")
      .text(function(d) { return d.name });

//get it going!
 force
      .nodes(nodes)
      .links(links)
      .start();

  force.on("tick", function() {


    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; });

//I think that translate changes all of the x and ys at once instead of one by
    one?
    node.attr("transform", function(d) { return "translate(" + d.x + "," + d.y +
        ")"; });

  })
  });
  });
</script>
```



Included Files:

config.py, getfollowers.py, iHussamFOLLOWERS.txt