INTRODUCTION TO WEB SCIENCES: Assignment 6

Manoj Chandra Kompalli

17 March 2016

Contents

1 Question 1:

1.1 Approach

I had 54 followers to start with. I was really confused how to begin but then I realized that all I needed to do was generate a JSON with two arrays. One with the nodes and other with the edges. Almost all the templates for d3 support this JSON format.

I could loop through my followers and can get their screen names and ids and can definitely show that they are connected to me, but the complicated task was showing the connections amongst themselves.

I decided to use Tweepy library. Tweepy would give me a Boolean value showing if the two people were connected or not. I had to write a function which would only take the names if they had atleast one true value.

That is, either the follower or followed by attribute is true. My next task was to generate the respective ids of the source and target screen names. I had then appended the data to my JSON array.

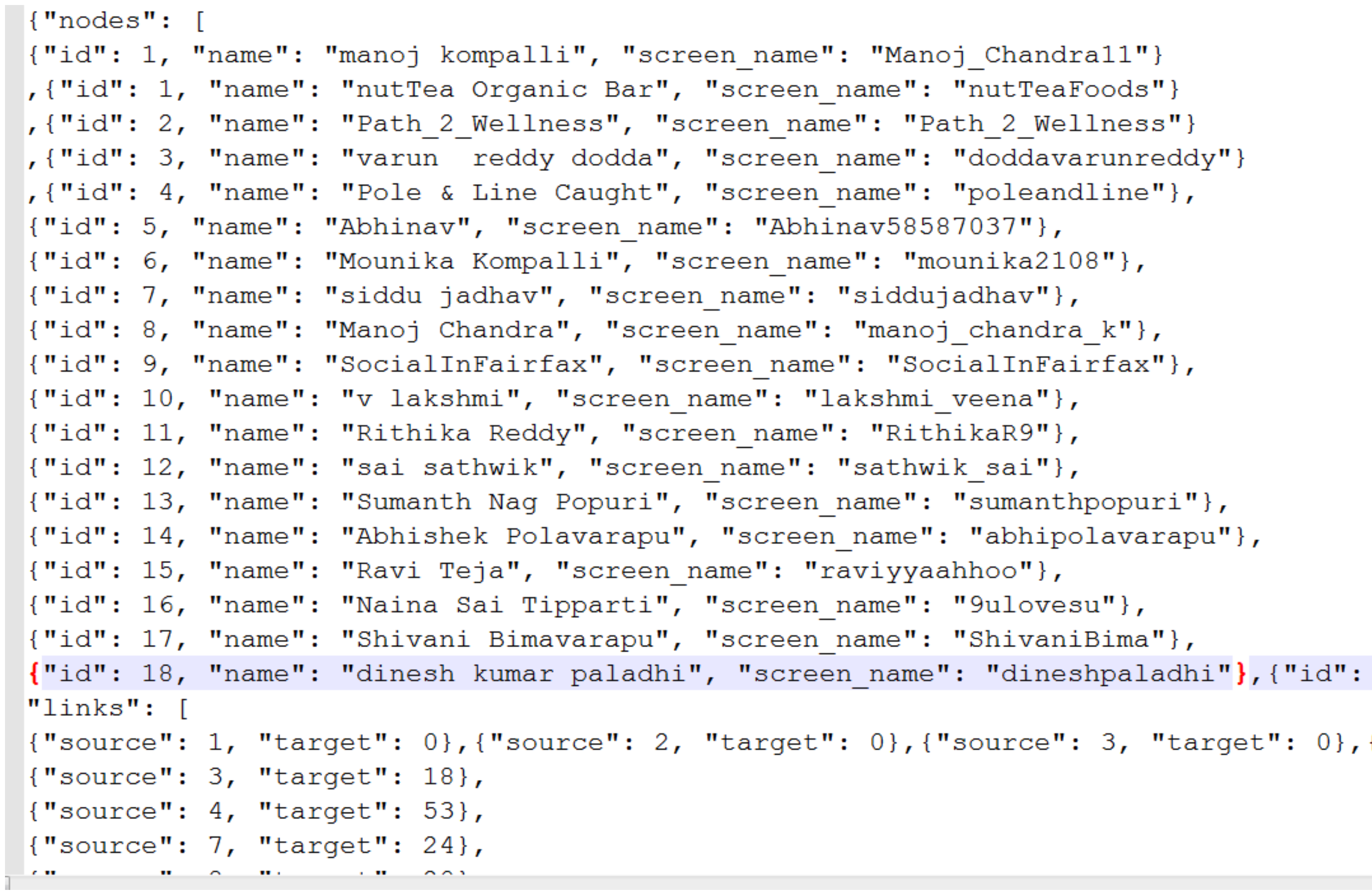
1.2 Code Listing

1.2.1 conections.py to generate the follower JSON

```python
import tweepy
import json
import time
import sys
import re

consumer_key="4b8YancHWbKNEH0TUacuIcWtV"
consumer_secret="SxEB55jiA0iqtbQFr2H18ey848nN5JVvfhkyamxC7qshLAC9oP"
access_token="54493821-U0YHBtEr6LsODZh6QizuhtL6eCOTwkZEuLtFH56Yu"
access_token_secret="hRTDURCuaOaG8Ri8XqCzBLJ0HlL4PhBDyf13gBqn95ZjH"
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)
file1=open('follower.json','r')
# file2=open('processed.json','w')
# file3=open('file3.json','a')
file4=open('file4.json','w')
file5=open('file5.json','w')
line=json.load(file1)
counter=0
result={}
output={}
my_dict={}
names={}
def screens():
  file3=open('file3.json','r')
  line1=json.load(file3)
  for each in line1:
    if(each['following']==True or each['followed_by']==True):
      names['source']=each['source']
      names['target']=each['target']

      file4.write(json.dumps(names)+',\n')

def ids():
  file4=open('file4.json','r')
  line1=json.load(file4)
  # for each in line1:
    # if each['source']==line['nodes'][x]['screen_name'] and each['target']==line['
    nodes'][x]['screen_name']
      # names['source']=line['nodes'][x]['screen_name']

    for x in range(0,54):
      for each in line1:
        source1=each['source']
        target1=each['target']

        source=line['nodes'][x]['screen_name']

        names['source']=line['nodes'][x]['id']

      for y in range(x,54):
        if(source==source1 or source==target1):
```

```python
        target=line['nodes'][y+1]['screen_name']
        output['target']=target




def function():
  file2=open('processed.json','r')
  line1=json.load(file2)
  for each in line1:
    source=each['source']
    target=each['target']
    con=api.show_friendship(source_screen_name=source,target_screen_name=target)
    my_dict['source']=con[0].screen_name
    my_dict['target']=con[1].screen_name
    my_dict['followed_by']=con[0].followed_by
    my_dict['following']=con[0].following
    file3.write(json.dumps(my_dict)+',\n')
  # for user in tweepy.Cursor(api.followers, screen_name="Manoj_Chandra11").items():
  # while True:
    # try:

      # counter+=1

def abc():
  for x in range(1,54):
    source=line['nodes'][x]['screen_name']
    output['source']=source

    for y in range(x,54):
      target=line['nodes'][y+1]['screen_name']
      output['target']=target
      file2.write(json.dumps(output)+',\n')




abc()
function()
screens()
```

1.3 Generated JSON file



```
{"nodes": [
{"id": 1, "name": "manoj kompalli", "screen_name": "Manoj_Chandra11"}
,{"id": 1, "name": "nutTea Organic Bar", "screen_name": "nutTeaFoods"}
,{"id": 2, "name": "Path_2_Wellness", "screen_name": "Path_2_Wellness"}
,{"id": 3, "name": "varun  reddy dodda", "screen_name": "doddavarunreddy"}
,{"id": 4, "name": "Pole & Line Caught", "screen_name": "poleandline"},
{"id": 5, "name": "Abhinav", "screen_name": "Abhinav58587037"},
{"id": 6, "name": "Mounika Kompalli", "screen_name": "mounika2108"},
{"id": 7, "name": "siddu jadhav", "screen_name": "siddujadhav"},
{"id": 8, "name": "Manoj Chandra", "screen_name": "manoj_chandra_k"},
{"id": 9, "name": "SocialInFairfax", "screen_name": "SocialInFairfax"},
{"id": 10, "name": "v lakshmi", "screen_name": "lakshmi_veena"},
{"id": 11, "name": "Rithika Reddy", "screen_name": "RithikaR9"},
{"id": 12, "name": "sai sathwik", "screen_name": "sathwik_sai"},
{"id": 13, "name": "Sumanth Nag Popuri", "screen_name": "sumanthpopuri"},
{"id": 14, "name": "Abhishek Polavarapu", "screen_name": "abhipolavarapu"},
{"id": 15, "name": "Ravi Teja", "screen_name": "raviyyaahhoo"},
{"id": 16, "name": "Naina Sai Tipparti", "screen_name": "9ulovesu"},
{"id": 17, "name": "Shivani Bimavarapu", "screen_name": "ShivaniBima"},
{"id": 18, "name": "dinesh kumar paladhi", "screen_name": "dineshpaladhi"},{"id":
"links": [
{"source": 1, "target": 0},{"source": 2, "target": 0},{"source": 3, "target": 0},
{"source": 3, "target": 18},
{"source": 4, "target": 53},
{"source": 7, "target": 24},
```

Figure 1: JSON array containing the follower data

1.4 Graph details

Here I used an avatar of a cat to show all of my followers. The directed graph has arrows directed from source to target. Example is if Yeshwanth follows Manoj, the edge is directed from Yeshwanth to Manoj.

1.5 Force directed graph of twitter followers

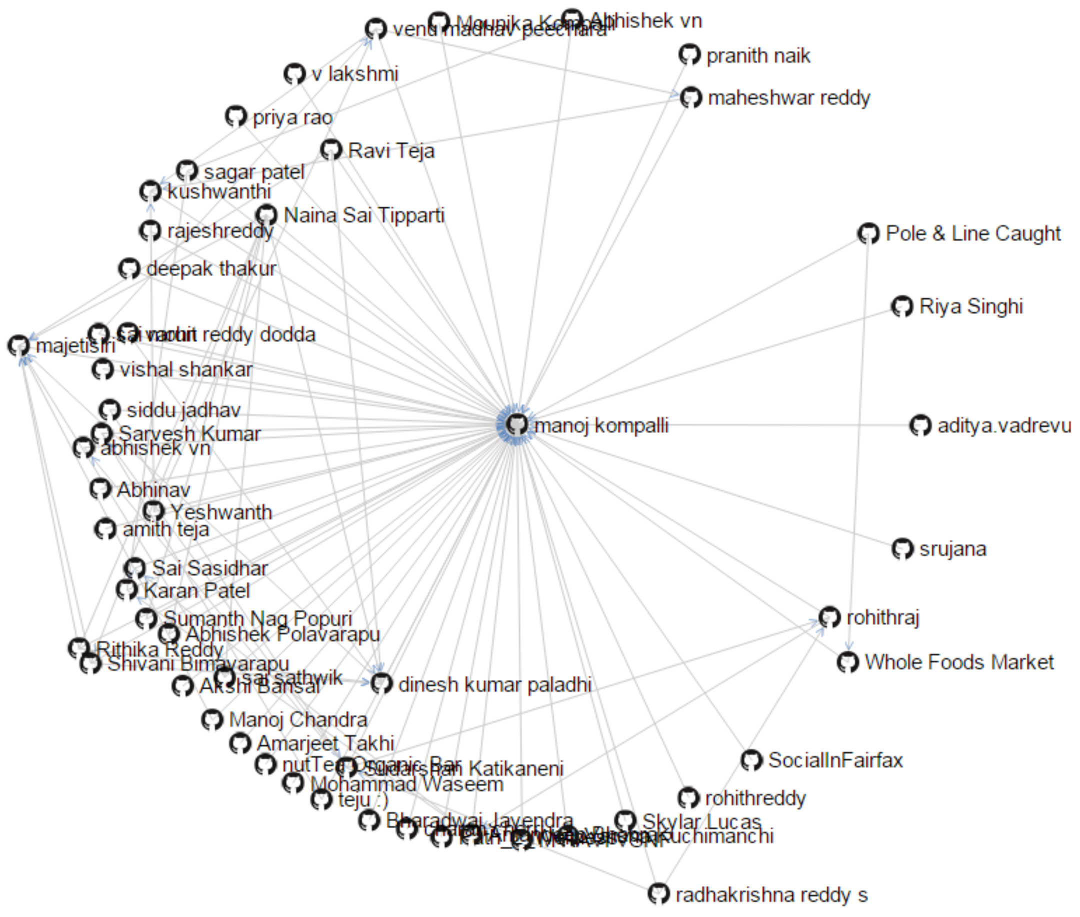


Figure 2: Twitter followers data of manoj kompalli

1.6 URI of graph listed in gist

`http://bl.ocks.org/manojchandrak/raw/5794f396fca3ab0dc5b7/`

2 Question 2:

2.1 Approach

Here, I had used the same JSON data as the above problem. I stripped off the first name from the name. I used urllib2 to run the genderize url on all the first names. I had generated JSON data which had all firstname along with their genders. I had integrated this data into the node data. I found some entries as null because they were labelled incorrectly or they dont belong to a person etc. I had generate the force directed graph from this data.

2.2 Code Listing

2.2.1 parse.py

```python
import re
import json
import sys
import urllib
import urllib2
import urlparse
url='https://api.genderize.io/?name='

details={}
file1=open('gender.json','w')
file2=open('follower.json','r')
line=json.load(file2)
file1.write('[\n')
for x in range(1,54):
  name=line['nodes'][x]['name'].split()[0]

  uri=url+name
  url_response = urllib2.urlopen(uri)

  file1.write(urllib2.urlopen(uri).read())
  file1.write(',\n')
file1.write(']')
def function():
  file1=open('gender.json','r')
  file2=open('follower.json','w')
  line1=json.load(file1)
  for x in range(1,54):
    gender=line1[x]['name']
    line[x]['gender']=gender
    file2.write(line[x]['gender'])

# function()
```

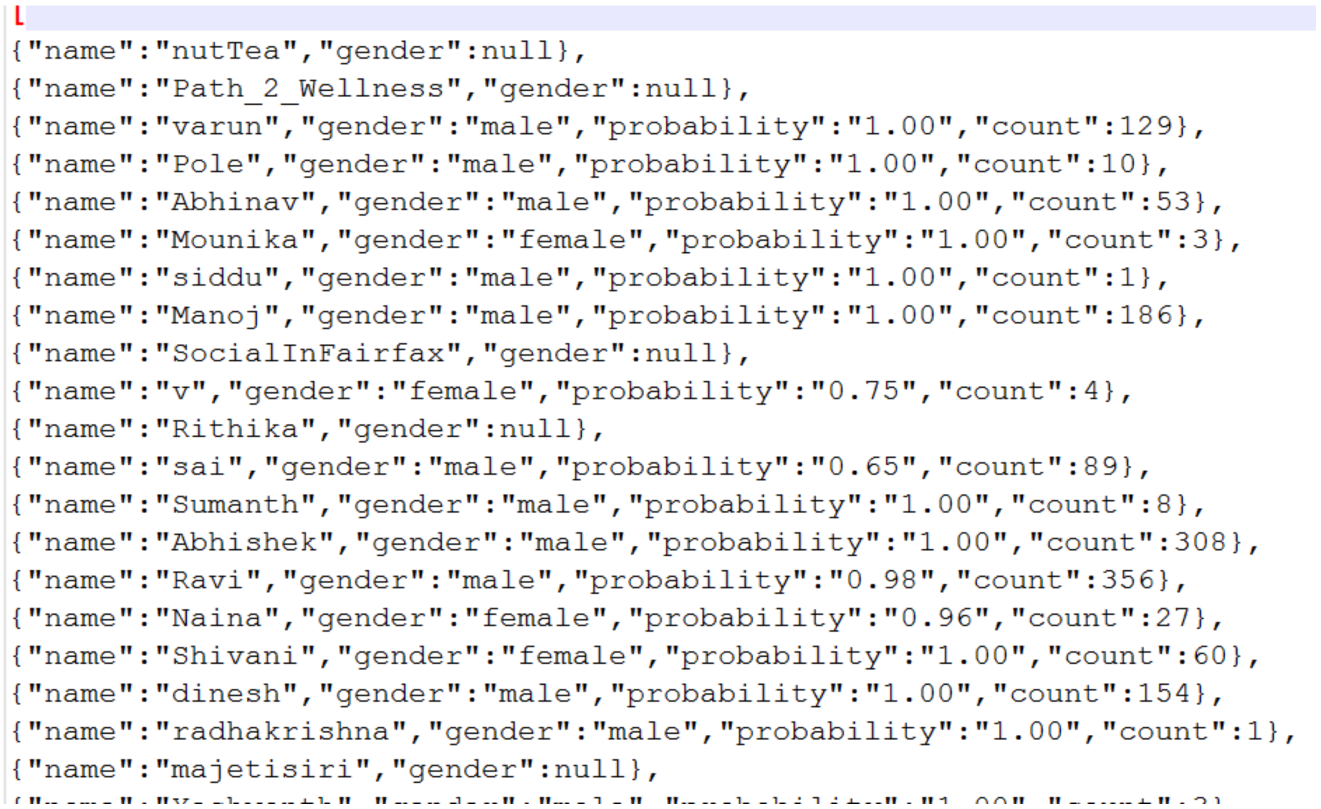
2.3 Code Listing

2.3.1 undirected.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>
.link {
  stroke: #ccc;
}
.node text {
  pointer-events: none;
  font: 15px sans-serif;
}
</style>
<body>
<script src="//d3js.org/d3.v3.min.js"></script>

<script>
var width = 1050,
    height = 960
var color = d3.scale.category20();
var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

var force = d3.layout.force()
    .gravity(0.05)
    .distance(300)
    .charge(-100)
    .size([width, height]);
d3.json("follower.json", function(error, json) {
  if (error) throw error;
  force
      .nodes(json.nodes)
      .links(json.links)
      .start();
  var link = svg.selectAll(".link")
      .data(json.links)
    .enter().append("line")
      .attr("class", "link")

  var node = svg.selectAll(".node")
      .data(json.nodes)
    .enter().append("circle")
      .attr("class", "node")
    .attr("r", 8)
    .style("fill", function (d) {
    return color(d.gender);
})
      .call(force.drag);
  node.append("circle")

      .attr("x", -8)
      .attr("y", -8)
      .attr("width", 16)
      .attr("height", 16);
  node.append("text")
      .attr("dx", 12)
      .attr("dy", ".35em")
      .text(function(d) { return d.name });
node.append("title")
      .text(function(d) { return d.name; });



  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; });
    node.attr("transform", function(d) { return "translate(" + d.x + "," + d.y + ")";
   });
  });
});
</script>
```

2.4 Parsed data

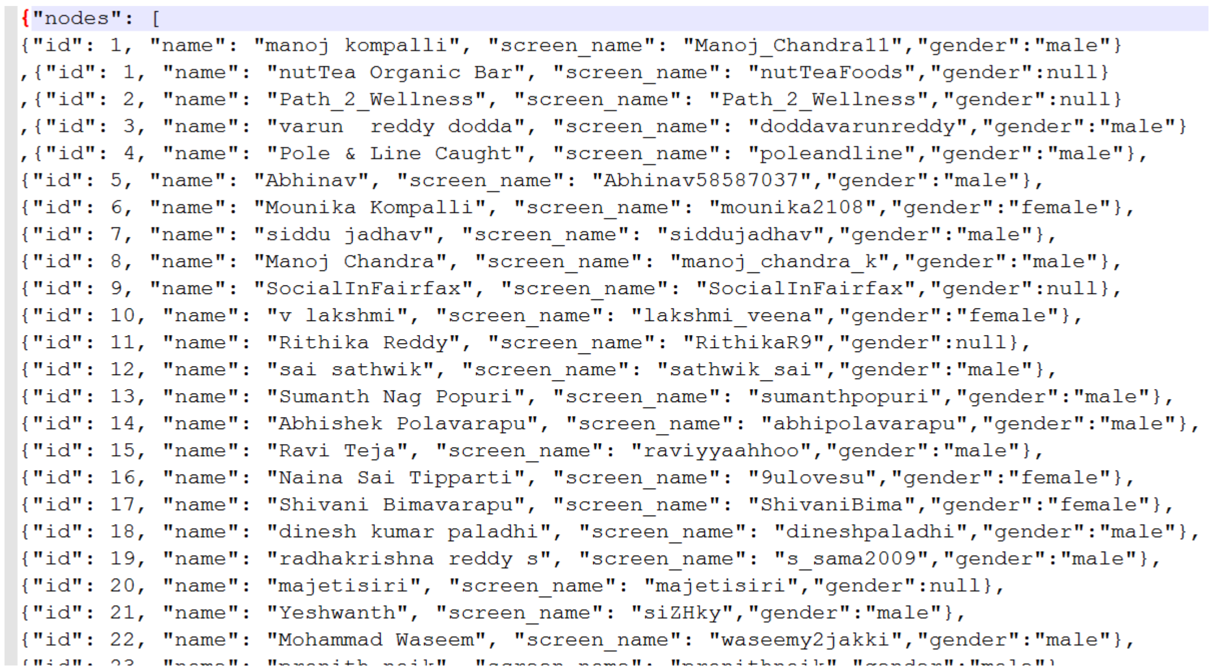
2.4.1 JSON response based on first name



```
[
{"name":"nutTea","gender":null},
{"name":"Path_2_Wellness","gender":null},
{"name":"varun","gender":"male","probability":"1.00","count":129},
{"name":"Pole","gender":"male","probability":"1.00","count":10},
{"name":"Abhinav","gender":"male","probability":"1.00","count":53},
{"name":"Mounika","gender":"female","probability":"1.00","count":3},
{"name":"siddu","gender":"male","probability":"1.00","count":1},
{"name":"Manoj","gender":"male","probability":"1.00","count":186},
{"name":"SocialInFairfax","gender":null},
{"name":"v","gender":"female","probability":"0.75","count":4},
{"name":"Rithika","gender":null},
{"name":"sai","gender":"male","probability":"0.65","count":89},
{"name":"Sumanth","gender":"male","probability":"1.00","count":8},
{"name":"Abhishek","gender":"male","probability":"1.00","count":308},
{"name":"Ravi","gender":"male","probability":"0.98","count":356},
{"name":"Naina","gender":"female","probability":"0.96","count":27},
{"name":"Shivani","gender":"female","probability":"1.00","count":60},
{"name":"dinesh","gender":"male","probability":"1.00","count":154},
{"name":"radhakrishna","gender":"male","probability":"1.00","count":1},
{"name":"majetisiri","gender":null},
```

Figure 3: Initial graph split into 4 groups

2.4.2 Follower data with gender details



```
{"nodes": [
{"id": 1, "name": "manoj kompalli", "screen_name": "Manoj_Chandra11","gender":"male"}
,{"id": 1, "name": "nutTea Organic Bar", "screen_name": "nutTeaFoods","gender":null}
,{"id": 2, "name": "Path_2_Wellness", "screen_name": "Path_2_Wellness","gender":null}
,{"id": 3, "name": "varun  reddy dodda", "screen_name": "doddavarunreddy","gender":"male"}
,{"id": 4, "name": "Pole & Line Caught", "screen_name": "poleandline","gender":"male"},
{"id": 5, "name": "Abhinav", "screen_name": "Abhinav58587037","gender":"male"},
{"id": 6, "name": "Mounika Kompalli", "screen_name": "mounika2108","gender":"female"},
{"id": 7, "name": "siddu jadhav", "screen_name": "siddujadhav","gender":"male"},
{"id": 8, "name": "Manoj Chandra", "screen_name": "manoj_chandra_k","gender":"male"},
{"id": 9, "name": "SocialInFairfax", "screen_name": "SocialInFairfax","gender":null},
{"id": 10, "name": "v lakshmi", "screen_name": "lakshmi_veena","gender":"female"},
{"id": 11, "name": "Rithika Reddy", "screen_name": "RithikaR9","gender":null},
{"id": 12, "name": "sai sathwik", "screen_name": "sathwik_sai","gender":"male"},
{"id": 13, "name": "Sumanth Nag Popuri", "screen_name": "sumanthpopuri","gender":"male"},
{"id": 14, "name": "Abhishek Polavarapu", "screen_name": "abhipolavarapu","gender":"male"},
{"id": 15, "name": "Ravi Teja", "screen_name": "raviyyaahhoo","gender":"male"},
{"id": 16, "name": "Naina Sai Tipparti", "screen_name": "9ulovesu","gender":"female"},
{"id": 17, "name": "Shivani Bimavarapu", "screen_name": "ShivaniBima","gender":"female"},
{"id": 18, "name": "dinesh kumar paladhi", "screen_name": "dineshpaladhi","gender":"male"},
{"id": 19, "name": "radhakrishna reddy s", "screen_name": "s_sama2009","gender":"male"},
{"id": 20, "name": "majetisiri", "screen_name": "majetisiri","gender":null},
{"id": 21, "name": "Yeshwanth", "screen_name": "siZHky","gender":"male"},
{"id": 22, "name": "Mohammad Waseem", "screen_name": "waseemy2jakki","gender":"male"},
```

Figure 4: Follower data with gender details

2.4.3 Graph details

I have figured out that my data had more males which itself is a proof for gender homophily. The blue dots represent the males, the light blue are the nulls and the orange ones represent females. There are only 7 orange nodes and most of them do not have many connections from my followers. This proves gender homophily

2.4.4 Force directed graph showing gender homophily

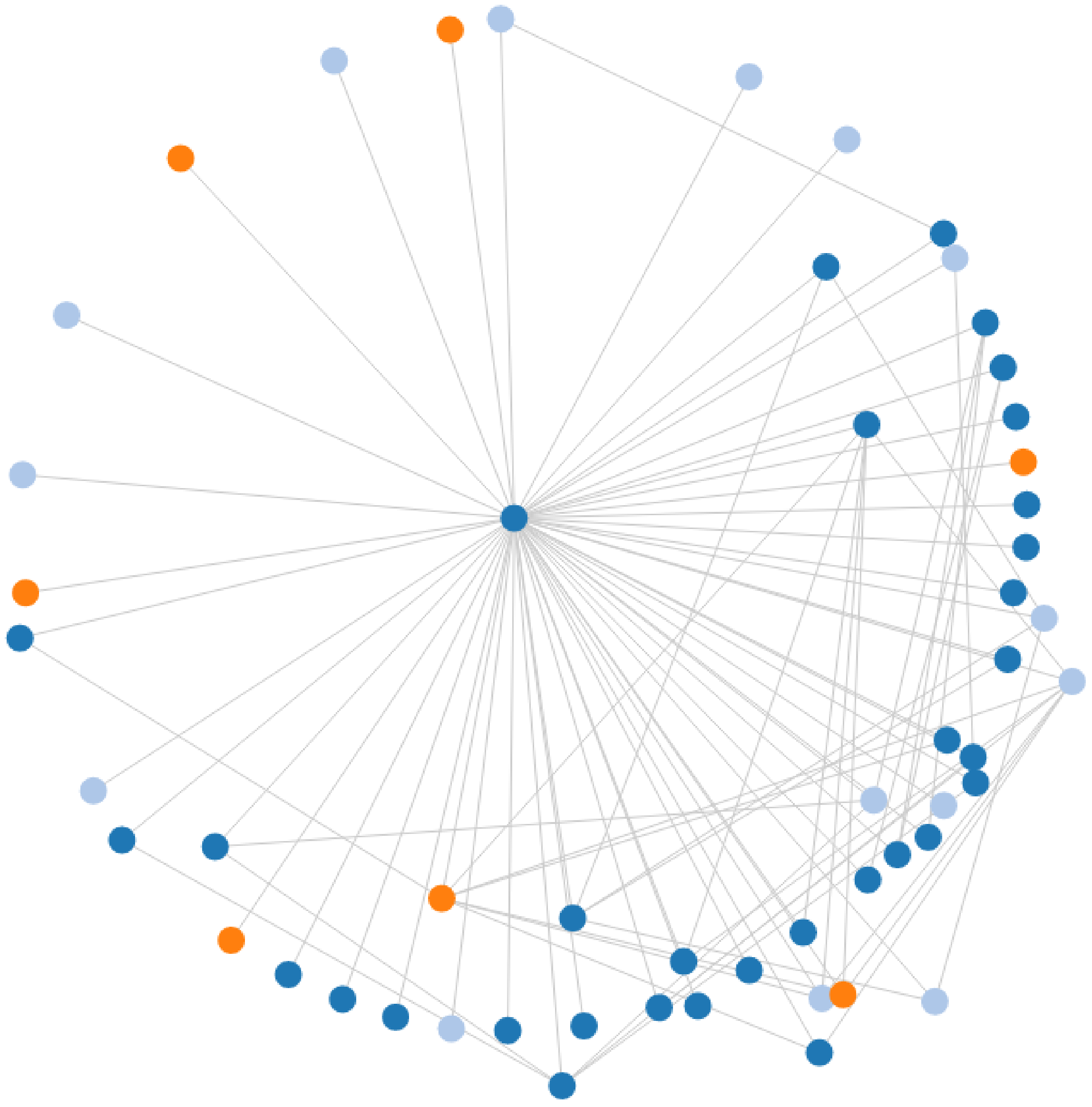


Figure 5: Force directed graph grouped on gender basis

2.5 URI of graph listed in gist

`http://bl.ocks.org/manojchandrak/raw/96b6e197ac4101f83446/`

Table 1: Table with followers and gender

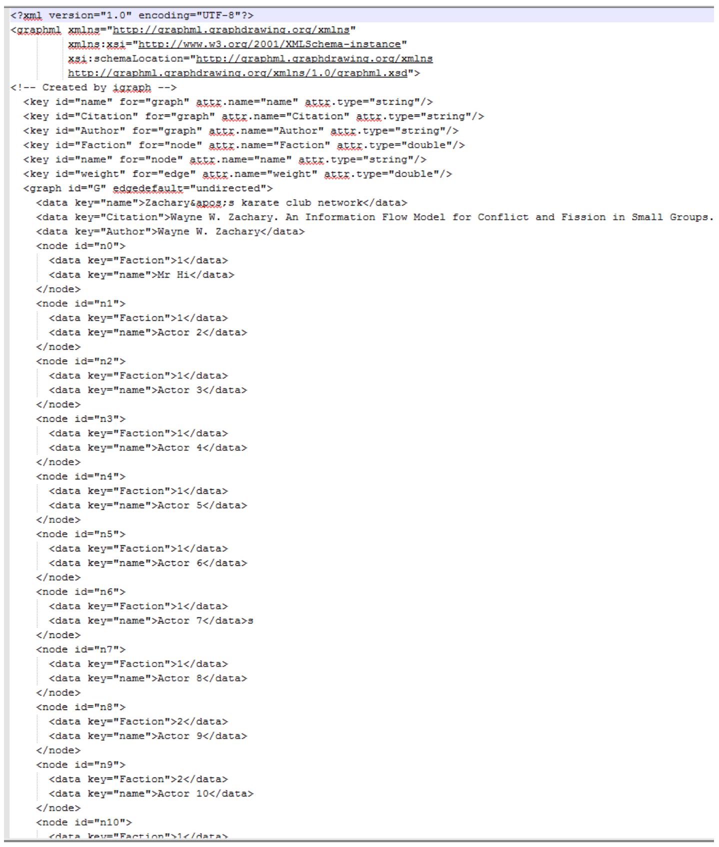
Name — Gender
nutTea — null
Path2Wellness — null
varun — male
Pole — male
Abhinav — male
Mounika — female
siddu — male
Manoj — male
SocialInFairfax — null
v— female
Rithika — null
sai — male
Sumanth — male
Abhishek — male
Ravi — male
Naina — female
Shivani — null
dinesh — male
radhakrishna — male
majetisiri — null
Yeshwanth — male
Mohammad — male
pranith — male
Sudarshan— male
rohithraj— null
Amandeep— male
charan— male
sai— male
M— male
rajeshreddy— null
teju‘— null
Sarvesh— male
rohithreddy— null
abhishek— male

3 Question 3:

3.1 Approach

I used the graphML input to generate a JSON response containing array of nodes and links. I had id, faction, name in the nodes array and source, target in the links array. I used this JSON to generate the force directed graph. Initially I did not group the members into their groups. I used a drop down which can toggle between split and no split so that the users can select if they want to see the entire club or the club after split. After split the two groups can be differentiated based on the color. On mouse hover, we can see the actors and the leader

3.2 Input data



```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
         http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<!-- Created by igraph -->
  <key id="name" for="graph" attr.name="name" attr.type="string"/>
  <key id="Citation" for="graph" attr.name="Citation" attr.type="string"/>
  <key id="Author" for="graph" attr.name="Author" attr.type="string"/>
  <key id="Faction" for="node" attr.name="Faction" attr.type="double"/>
  <key id="name" for="node" attr.name="name" attr.type="string"/>
  <key id="weight" for="edge" attr.name="weight" attr.type="double"/>
  <graph id="G" edgedefault="undirected">
    <data key="name">Zachary&apos;s karate club network</data>
    <data key="Citation">Wayne W. Zachary. An Information Flow Model for Conflict and Fission in Small Groups.
    <data key="Author">Wayne W. Zachary</data>
    <node id="n0">
      <data key="Faction">1</data>
      <data key="name">Mr Hi</data>
    </node>
    <node id="n1">
      <data key="Faction">1</data>
      <data key="name">Actor 2</data>
    </node>
    <node id="n2">
      <data key="Faction">1</data>
      <data key="name">Actor 3</data>
    </node>
    <node id="n3">
      <data key="Faction">1</data>
      <data key="name">Actor 4</data>
    </node>
    <node id="n4">
      <data key="Faction">1</data>
      <data key="name">Actor 5</data>
    </node>
    <node id="n5">
      <data key="Faction">1</data>
      <data key="name">Actor 6</data>
    </node>
    <node id="n6">
      <data key="Faction">1</data>
      <data key="name">Actor 7</data>s
    </node>
    <node id="n7">
      <data key="Faction">1</data>
      <data key="name">Actor 8</data>
    </node>
    <node id="n8">
      <data key="Faction">2</data>
      <data key="name">Actor 9</data>
    </node>
    <node id="n9">
      <data key="Faction">2</data>
      <data key="name">Actor 10</data>
    </node>
    <node id="n10">
      <data key="Faction">1</data>
```

Figure 6: GraphML data of karate club problem

3.3 Code Listing

3.3.1 convert.py

```
#!/usr/bin/env python

import sys
from bs4 import BeautifulSoup
counter =33
file1 = open('karate.GraphML', 'r')
file2 = open('output1.json', 'w')
soup     = BeautifulSoup(file1 , "html5lib")
file2.write('"nodes": [\n' )
for node in soup.find_all('node'):
        nodekeys = dict(node.attrs)
        id  = nodekeys[u'id']
        data_faction , data_name   = node.find_all('data')

        faction = data_faction.contents
        name    = data_name.contents

        id_split = id.split('n');
        counternt           = id_split[1]

        file2.write('  {\n')
        file2.write('   "id": ')
        file2.write( id_split[1])
        file2.write(',\n')

        file2.write('   "faction": ')
        file2.write(faction[0])
        file2.write(',\n')

        file2.write('   "name": "')
        file2.write(name[0])
        file2.write('"\n ')

        if counternt != 33:
            file2.write(' },\n')
        else :
            file2.write(' }\n ],')


file2.write('\n "links": [\n' )
for edge in soup.find_all('edge'):
        edgekeys  = dict(edge.attrs)
        source = edgekeys[u'source']
        target = edgekeys[u'target']

        source_split = source.split('n')
        target_split = target.split('n')

        file2.write('  {\n')
        file2.write('   "source": ')
        file2.write(source_split[1])
        file2.write(',\n')

        file2.write('    "target": ')
        file2.write(target_split[1])
        file2.write('},\n')



file2.write(']} ')
```

3.4 Output

3.4.1 Without Split

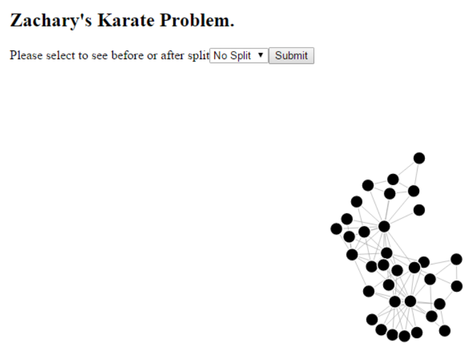

Zachary's Karate Problem.
Please select to see before or after split
No Split
Submit

Figure 7: Karate group without split

3.4.2 Karate group with split

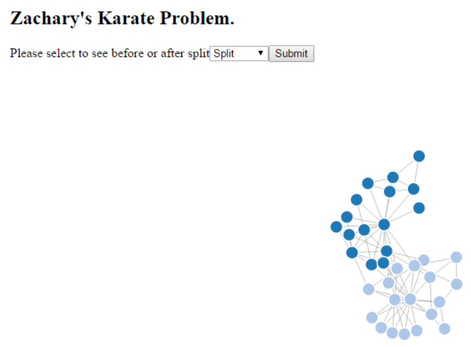

Zachary's Karate Problem.
Please select to see before or after split
Split
Submit

Figure 8: Karate group with split

3.5 URI of graph listed in gist

http://bl.ocks.org/manojchandrak/raw/50e8ccde5a7e8bc2d5c4/

References

[1] For Graph layouts: `https://github.com/mbostock/d3/wiki/Force-Layout`.

[2] For converting to directed graph : `http://www.coppelia.io/2014/07/an-a-to-z-of-extra-features-for-the-d3-force-layout/`.

[3] To extract gender data : `https://genderize.io/`.

[]