Visualization

Visualizing Clustering

Here, I use the dataset I used in cluster_with_real_data and visualize the data based on ground-truth categories, and show it in an HTML webpage. Also, I use the clustering result of my cluster_with_real_data and visualize the data based on my clustering, and show it in an HTML webpage.

```
<!DOCTYPE html>
<html>
<head>
	<title> </title>
</head>
<body>
	<h1> Assignment4 Q1 </h1>
	<script src="https://d3js.org/d3.v4.min.js"></script>
	<script src='https://rawgit.com/karpathy/tsnejs/master/tsne.js'></script>
	<script type="text/javascript">
		// Parameters
		const width = 960,
			height = 900,
			margin = 40,
			colorScale = d3.scaleOrdinal(d3.schemeCategory20),
			centerx = d3.scaleLinear()
				.range([width / 2 - height /2 + margin, width /2 + height /2 -
margin]),
			centery = d3.scaleLinear()
				.range([margin, height - margin]);

		// Draw nodes
		function draw(canvas, nodes){
			let context = canvas.node().getContext("2d");
			context.clearRect(0, 0, width, width);

			for(var i = 0, n = nodes.length; i < n; ++i) {
				var node = nodes[i];
				context.beginPath();
				context.moveTo(node.x, node.y);
				context.arc(node.x, node.y, 20, 0, 2 * Math.PI);

				context.lineWidth = 0.5;
				context.fillStyle = node.color;
				context.fill();

				context.fillStyle = "black";
```

```
                        context.font = "20px Open Sans";
                        context.fillText(node.text, node.x, node.y);
                  }
            }

            // Load data
            d3.csv('df.csv', function (src_data){
                  // Create t-SNE model
                  const data = src_data.map((d, i) => Object.values(d));

                  const canvas = d3.select("body").append("canvas")
                        .attr("width", width)
                        .attr("height", height);

                  const model = new tsnejs.tSNE({
                        dim: 2,
                        perplexity: 30,
                  });

                  var features = data.map(function(value, index) {return
value.slice(1,-1);});
                  model.initDataRaw(features);


                  // Create force simulation
                  const forcetsne = d3.forceSimulation(
                        data.map(
                              d => (d.x = width / 2, d.y = height / 2, d)
                        )
                  )
                  .alphaDecay(0.005)
                  .alpha(0.1);

                  // Add force (Update nodes' positions)
                  forcetsne.force('tsne', function (alpha){
                        model.step();

                        let pos = model.getSolution();

                  centerx.domain(d3.extent(pos.map(d => d[0])));
                  centery.domain(d3.extent(pos.map(d => d[1])));

                  data.forEach((d,i) => {
                        d.x += alpha * (centerx(pos[i][0]) - d.x);
                        d.y += alpha * (centery(pos[i][1]) - d.y);
                  });
            })
```

```
            .force('collide', d3.forceCollide().radius(d => 10));

            // Draw nodes
            forcetsne.on('tick', function(){
                  let nodes = data.map((d,i) => {
                        return {
                              x: d.x,
                              y: d.y,
                              color: colorScale(d[d.length - 1]),
                              text: d[0]
                        };
                  });

                  draw(canvas, nodes);
            });
      })


      </script>
</body>
</html>
```

In order to apply the code above to the raw data and clustering results in HW3 Q1, we need to do some cleaning to our data as below:

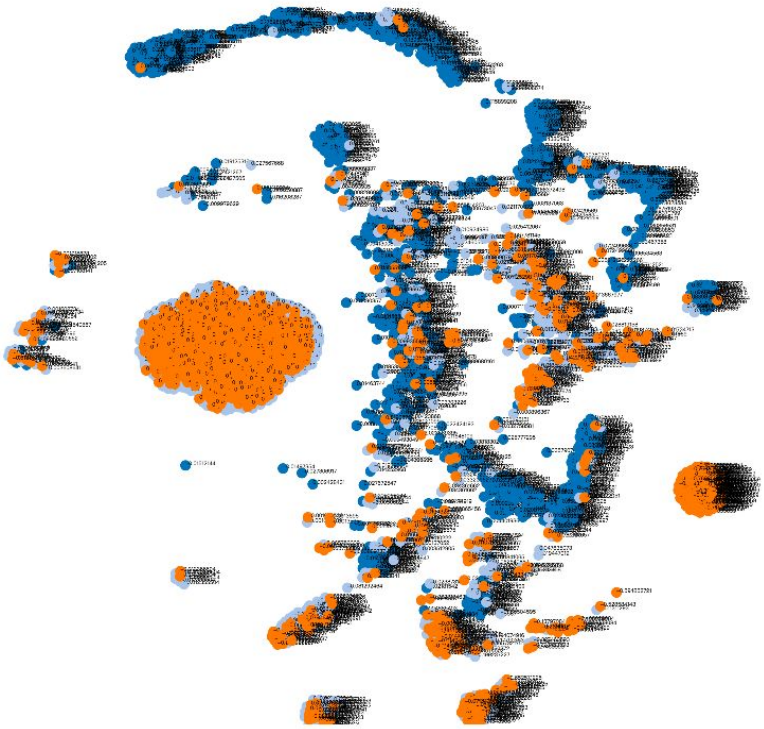
```
import numpy as np

from pyspark.sql.types import *

a = udf(lambda x: x.toArray().tolist(), ArrayType(DoubleType()))
c = predict_error.select(a('features').alias('features')).collect()
d = pd.DataFrame(list(map(lambda x:x.features,c)))

predict_error_df = predict_error.toPandas()
df = pd.concat([predict_error_df, d], axis=1)
df = df.drop(['features'], axis=1)
df = df.drop(['error'], axis = 1)
df.to_csv('./df.csv')
```

```
d.head()
```

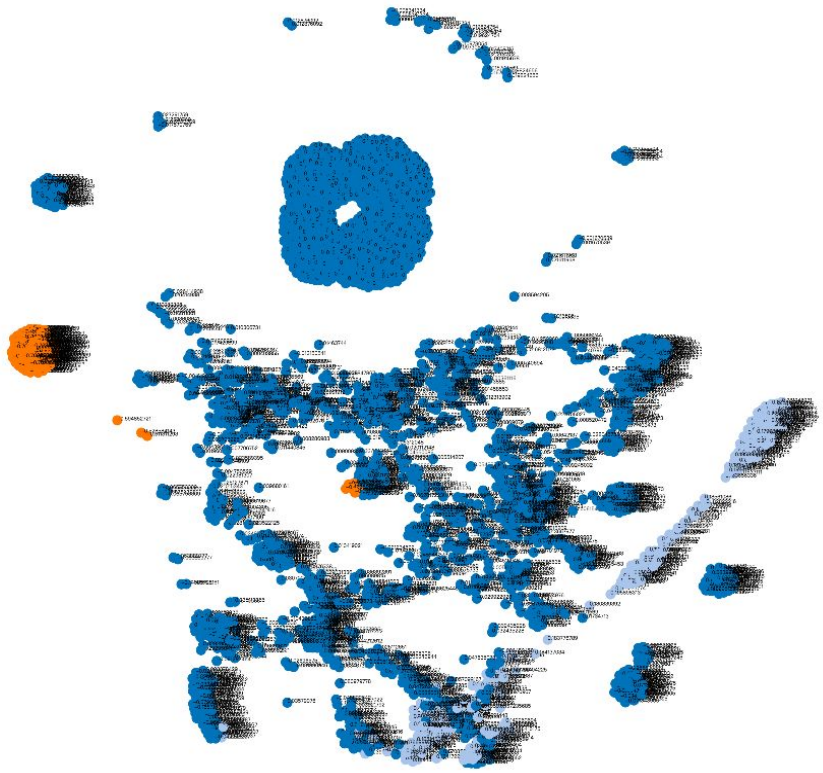
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
0	0.007010	-0.019289	0.025946	0.013056	0.003419	0.004882	-0.010775	-0.021567	0.002658	0.006513	-0.014982	-0.014949	0.006320	-0.001909	-0.00
1	0.155865	-0.142317	0.019319	-0.092397	0.058404	0.124724	0.095632	0.260105	0.192751	0.144389	0.047203	-0.023474	-0.001754	-0.015176	-0.05
2	-0.077433	0.096169	0.037499	-0.034897	0.036292	-0.073900	-0.050947	-0.141195	-0.173361	-0.055793	-0.027058	0.009056	0.006837	-0.073317	0.02
3	0.013015	0.027059	-0.021568	0.005518	-0.011345	0.011553	0.016978	-0.002686	0.010743	-0.010278	-0.003045	-0.024358	0.009056	0.018320	0.00
4	0.013018	-0.013640	0.012951	0.023998	0.001702	0.003539	-0.006839	-0.001637	0.006672	0.000218	0.015409	-0.006203	-0.025777	0.010106	0.03

```
jbiandeMacBook-Air:hw4 jbian$ python3 -m http.server 5000
```

We could get the plots as below:



Visualization of Raw Data

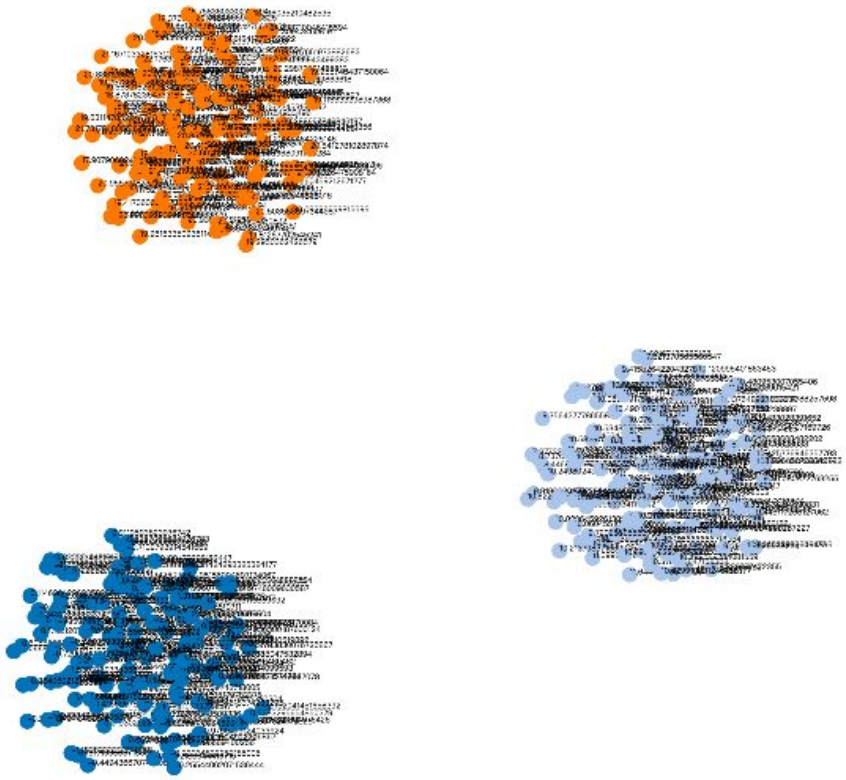


Visualization of Clustering Results

However, we could see that the results are not good, in order to illustrate our d3 code is correct, we randomly generate data from three gaussian clusters.

```
import numpy as np
import pandas as pd
data1 = np.random.multivariate_normal([0]*20, np.identity(20),150)
data2 = np.random.multivariate_normal([10]*20, np.identity(20),150)
data3 = np.random.multivariate_normal([20]*20, np.identity(20),150)
data = np.vstack((data1, data2, data3))
data_df = pd.DataFrame(data = data)
label1 = np.zeros(150)
label2 = label1 + 1
label3 = label2 + 1
label = np.hstack((label1, label2, label3))
data_df['label'] = label
data_df.to_csv('./random_data.csv', index = False)
```

And the visualization of the randomly generating results are as below:



Visualization of Random Generated Gaussian data

Visualizing Streams

I show the real-time stream of Tweets that match 10 hashtags I specified with a bar chart including number and name for each bar.

```
jb4076@big-data-analytics:wget -qO- https://deb.nodesource.com/setup_8.x | sudo -E bash
-
jb4076@big-data-analytics:sudo apt-get install -y nodejs
jb4076@big-data-analytics:~/hw4$ npm install
jb4076@big-data-analytics:~/hw4$ node app.js
```

Codes from index.html are as below:

```
<!DOCTYPE html>
    <html>
    <head>
        <meta charset = "utf-8">
        <meta http-equiv="X-UA-Compatible" content= "IE=edge">
        <title>Twitter Hashtag Monitoring with Node JS, Socket.io, Express,
ntwitter</title>
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.js"></script>
        <script src="/socket.io/socket.io.js"></script>
```

```
    <script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
    <script type="text/javascript">
        var track_item = ["#nyc","#data","#Job","#Trump",
                          "#Movies","#song","#holiday",
                          "#Friday","#music","#Halloween"];
        var count = {}
        track_item.forEach(function(term){
            count[term] = 0;
        });
        var w = 600;
        var h = 600;
        var barPadding = 1;
        var svg = d3.select("body")
            .append("svg")
            .attr("width",w)
            .attr("height",h);


        var dataset = Object.keys(count).map(function(key){
            return count[key];

        });

         svg.selectAll("rect")
           .data(dataset)
           .enter()
           .append("rect")
           .attr("x",function(d,i){
               return i * (w /dataset.length);
           })
           .attr("y",function(d){
               return h - (d*4);
           })
           .attr("width",w / dataset.length-barPadding)
           .attr("height",function(d){
               return d*4;
           })
           .attr("fill",function(d){
            return "rgb(0, 0, " + (d * 10)+ ")";
           });
         svg.selectAll("text")
            .data(dataset)
            .enter()
            .append("text")
            .text(function(d, i){
```

```
            return [d, track_item[i]];
        })
        .attr("text-anchor", "middle")
        .attr("x",function(d,i){
            return i * (w/dataset.length) * (w/dataset.length-barPadding)/2;
        })
        .attr("y", function(d){
            return h;
        })
        .attr("font-family","sans-serif")
        .attr("font-size","14px")
        .attr("fill","blue");



$(document).ready(function(){
    var socket = io.connect("http://35.196.64.46:5000/");
    socket.on("stream", function(data){
    console.log(count)

    count[data.hashtag] += 1;
     var dataset = Object.keys(count).map(function(key){
          return count[key];
    });

    svg.selectAll("rect")
       .data(dataset)
       .transition()
       .attr("x", function(d,i){
           return i * (w/dataset.length);
       })
       .attr("y", function(d){
           return h - (d*4);
       })
       .attr("width", w/dataset.length - barPadding)
       .attr("height",function(d){
           return d*4;
       })
       .attr("fill", function(d){
           return "rgb(0, 0, " + (d*10) + ")";
       });

     svg.selectAll("text")
        .data(dataset)
        .transition()
        .text(function(d, i){
            return [d, track_item[i]];
```

```
                    })
                    .attr("text-anchor", "middle")
                    .attr("x",function(d,i){
                         console.log(i)
                         return i * (w/dataset.length) +
                                   (w/dataset.length-barPadding)/2;
                    })
                    .attr("y",function(d,i){
                          return h - (d*4);
                    })
                    .attr("font-family","sans-serif")
                    .attr("font-size","14px")
                    .attr("fill","black");
                    });

                });
            </script>
            <ul id = "tweets"></ul>
    </body>
    </html>
```

Codes from app.js are as below:

```
var twitter = require('twit'),
credentials = require('./credentials.js'),
express = require('express'),
app = express(),
server = require('http').createServer(app),
io = require('socket.io').listen(server);

var t = new twitter({
      consumer_key: credentials.consumer_key,
      consumer_secret: credentials.consumer_secret,
      access_token: credentials.access_token_key,
      access_token_secret: credentials.access_token_secret
});

app.get('/', function (req, res){
            res.sendFile(__dirname + '/index.html');
});
server.listen(5000);

function containsAny(str, substrings) {
      for (var i = 0; i != substrings.length; i++) {
            var substring = substrings[i];
            if (str.indexOf(substring) != -1){
```

```
                  return substring;
            }
      }
      return null;
}
io.sockets.on('connection', function (socket){
      console.log('SOCKET CONNECTED\n');
      var track_item = ["#nyc","#data","#Job","#Trump",
                        "#Movies","#song","#holiday",
                        "#Friday","#music","#Halloween"];
      var twitterStream = t.stream(
            'statuses/filter',
            { track: track_item }
      );
      twitterStream.on('tweet', function(tweet){
            matchedHashtag = containsAny(tweet.text, track_item)
            if(tweet && matchedHashtag) {
                  console.log(tweet.text)
                  io.sockets.emit('stream', { detail: tweet, hashtag: matchedHashtag
});
            }
      });
      twitterStream.on('error', function(error){
            throw error;
      });
});
```

Codes from credentials.js are as below:

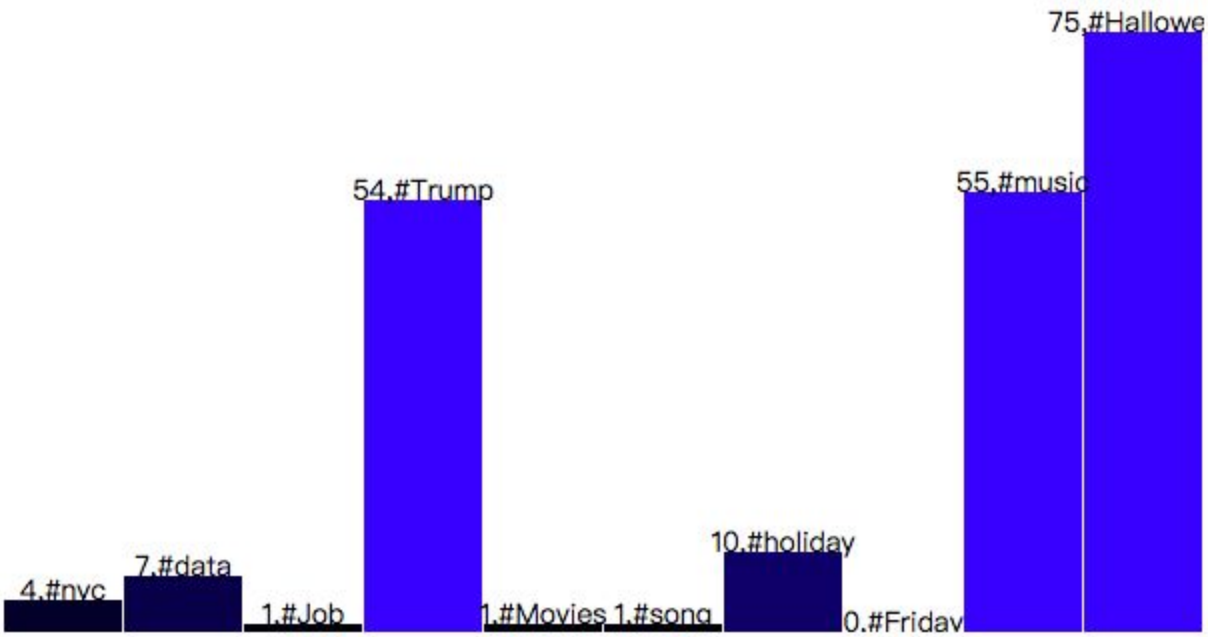
```
module.exports = {
      "consumer_key": "k3AOTJKE13xBW9gkBd3TATY9X",
      "consumer_secret": "kGN9HEL4HALvLFMyhmWYE2xThgzlBwNSLeSxdpc31LLXDoQJS6",
      "access_token_key": "1011265741444845568-htdQhk4GmHZn40evy7sJU0mOO4vmmD",
      "access_token_secret": "JYlN8k961i3GSmGgzriDhmh7undn4BHVQczN5SjyvcBkx"
}
```

Codes from package.json are as below:

```
{
      "name": "twitter-hashtag-monitor",
      "description": "monitor hastage in realtime.",
      "version": "0.0.1",
      "private": true,
      "dependencies": {
            "socket.io": "*",
```

```
            "express": "*",
            "twit": "^2.2.5"
        }
}
```

Plot on webpage are as below:



Tweet text are as below:

```
@Briteeye777
@ColoradoGirl2A
@di_plora
@ElleHart2Hart
@MissILmom
@AngeleSta…
Posted a new song: "GsmgBeats AllBlack 2018" https://t.co/ffVV6IIZ0W #music https://t.co/QTQGW9GqkW
Got back from celebrating the best holiday of them all. Now watching the fight. #Halloween https://t.co/SIEhigjxqW
RT @trianglar1230: Holiday MV❤️❤️❤️ It is very cute that Joshua is writing letters and I appreciate Wonwoo :)

#seventeen #joshua #holiday…
🔪 Stop Paying High #Movie #Download Fees ▶ News-Entertainment .org ◀ Pay ONCE for #High #Def UNLIMITED #Movies https://t.co/S8ZXXqygQl
RT @realAAnon: #Q #RedWave #WeThePeople #PatriotsUnited #Trump #DeepState #MAGA #VoteRedToSaveAmerica #Deplorables #VoteRed #PatriotsFight…
#Trump Lawyer Rudy #Giuliani Takes Mystery Trips to #Russia, Armenia and Ukraine https://t.co/1v0KmvJQRR
RT @sewwutnow: #BuildTheWall, Concertina Wire Wall‼God bless our military‼God bless @POTUS @realDonaldTrump #Trump for keeping America sa…
Promote your #edm #music and #musicvideo on #ArtistRack the number 1 music blog for #unsigned #indie #artist… https://t.co/18Ist94VJX
RT @kimfaul: Happy Happy #Halloween,
Halloween,
Halloween!

Happy Happy Halloween,
Silver Shamrock! 🍀🍊

#LEGO #HalloweenIII #SeasonOfTh…
RT @Tennesseine: Every time #Obama golfed or even went out to dinner, #Republicans howled at the expense. #Trump, who famously said he'd NE…
RT @SirenTV: Anyone going as a mermaid this year?
Share your costumes with us. #Halloween #Siren
RT @funamusea: #Halloween https://t.co/zRYrWWDMap
RT @vilmavargasva: #Halloween  #SofíaEspín #FernandoAlvarado #NormaVallejo #RicardoPatiño #JorgeGlas https://t.co/Xmr5Iyw1qb
RT @spikedonline: Is #Trump to blame for pipe-bombs and anti-Semitic slaughter?

@Tom_Slater_ @Ella_M_Whelan and @FraserMyers discuss on t…
Number1 #music #blog #ArtistRack, you can promote your #rap #music and #musicvideos, submit here… https://t.co/MzRex9PLdh
@PradRachael @LindaWa88949972 Let's be clear. Nobody is advocating "open borders." That is a #Trump fear point. Be… https://t.co/jn8q1ve6vv
RT @Unkle_Ken: #Wisconsin conservative admits he'd shoot his sister in the face for #Trump: 'She has to know how passionate I am'

https:/…
RT @NightLord2009: "All #vampires are brothers in the brotherhood of #night! - #StewartStafford (@TheVorbing) #Halloween #Halloween2018movi…
RT @UserExperienceU: Amazing #Halloween outfit of #PostMalone by creative artist and designer #nda #nancydrewartist she makes hand painted…
RT @funamusea: #Halloween https://t.co/zRYrWWDMap
Chinga Chinga #Trump ya enfilando militares en la frontera y estos wey es no han salido de chiapas
Need some great #data to work with? Check out @Canada's Open Government Portal! 📊📈 https://t.co/ALD4swW0vH
RT @brendaoncats: Sweet darling baby, friendly to kids, INU NEEDS U now.  #cats #nyc https://t.co/AkeB0Euwno
RT @soteeeeeetu: ●
テロメアの産声 / Heavenz
covered by ●.*(そてつ)*.●
```

Visualizing Graphs

I created an HTML webpage which visualizes the graph I generalized in graph_analysis. I set a threshold on the maximum node numbers I want to display in your visualization. And I highlighted the nodes that have the large PageRank numbers in the graph with different color.

Firstly, the process of data generation is as below:

```
home/jb4076/spark/bin/pyspark --packages graphframes:graphframes:0.6.0-spark2.3-s_2.11
```

```
>>> from graphframes import *
>>> e = sqlContext.read.format("csv").option("header", "true").load("./hw3/e.csv")
>>> v = sqlContext.read.format("csv").option("header", "true").load("./hw3/v.csv")
>>> g = GraphFrame(v, e)
>>> g.edges.toPandas().to_csv('1.csv',sep=',',index = False, encoding = 'utf-8')
>>> results = g.pageRank(resetProbability=0.15, maxIter=10)
>>> results.vertices.select("id", "pagerank").show()
>>> results = g.pageRank(resetProbability=0.15, maxIter=10)
>>> results.toPandas().to_csv('pager.csv',sep=',',index = False, encoding = 'utf-8')
>>> sc.setCheckpointDir('.')
>>> results = g.connectedComponents()
>>> results.toPandas().to_csv('comp.csv',sep=',',index = False, encoding = 'utf-8')

>>> import pandas as pd
>>> edge = pd.read_csv('./1.csv')
>>> pagerank = pd.read_csv('./pager.csv')
>>> component = pd.read_csv('./comp.csv')
>>> edge_u = edge.drop(['dot', 'relationship'], axis = 1)
>>> edge_u.columns = ['source','target']
>>> edge_u.head()
```

```
source,target
41445,41457
15528,41477
15510,15529
41463,41483
41463,41504
15494,41484
41490,41504
15506,41445
41455,41504
41479,41483
15491,41478
15499,41490
15519,41483
41477,41484
15507,41437
41455,41464
15529,41445
41477,41483
15510,15528
15491,15523
15499,41478
15498,41483
15526,41504
15497,41452
41437,41445
15519,41455
41445,41486
15526,15528
15525,41486
```

```
>>> edge_u.to_csv('./edge.csv')
>>> node = pagerank[['id', 'pagerank']].merge(component[['id','component']])
>>> node.to_csv('./node.csv')
```

```
ID,PageRank,Component
15494,0.321220616,17179869184
15510,0.321220616,17179869184
15491,0.321220616,17179869184
15497,0.321220616,17179869184
15519,0.321220616,17179869184
15507,0.321220616,17179869184
15485,0.321220616,17179869184
15512,0.321220616,17179869184
15495,0.339423118,17179869184
15531,0.340454593,17179869184
15498,0.355350307,17179869184
15496,0.358657095,17179869184
15524,0.389479997,17179869184
15499,0.410376676,17179869184
41455,0.45882566,17179869184
15529,0.470495904,17179869184
15526,0.360225977,17179869184
15523,0.377753135,17179869184
15506,0.417266008,17179869184
15525,0.441971167,17179869184
15528,0.445387681,17179869184
41464,0.617227778,17179869184
41457,0.618414445,17179869184
41437,0.777805766,17179869184
41477,0.853181643,17179869184
41445,0.894791321,17179869184
41467,0.895561185,17179869184
41465,0.97439789,17179869184
41479,1.138952966,17179869184
```

Then, we plot it in a webpage:

```
<!DOCTYPE html>
<html>
<head>
	<title> Q3 </title>
</head>
<body>
	<h1> Q3 </h1>
	<script src="https://d3js.org/d3.v4.min.js"></script>
	<script src='https://rawgit.com/karpathy/tsnejs/master/tsne.js'></script>
	<script type="text/javascript">
		var width = 960, height = 600;

		var svg = d3.select("body").append("svg")
		    .attr("width", width)
		    .attr("height", height);
		var g = svg.append("g").attr("transform", "translate(" + width/2 + "," +
height/2 + ")");
		var radiusScale = d3.scaleLinear()
		    .domain([0,2])
		    .range([5,30])
```

```
            d3.csv('node1.csv',function(node_data){
                  d3.csv('edge1.csv',function(edge_data){
                        //Initialize force simulation
                        const nodes = node_data;
                        const links = edge_data.map(function(a){
                              return {source: Number(a.source), target:
Number(a.target)}
                        });


                        var simulation = d3.forceSimulation(nodes)
                            .force("charge", d3.forceManyBody().strength(-500))
                            .force("link", d3.forceLink(links).id(function(d, i){
                              return d.ID}).distance(20).strength(0.01).iterations(10))
                            .force("x", d3.forceX())
                            .force("y", d3.forceY())
                            .on('tick', tick);

                        //Set attributes for line and node
                        var lineG = g.append("g")
                              .attr("stroke", "#000")
                              .attr("stroke-width",1.5)
                            .selectAll("line")
                            .data(links)
                            .enter().append("line");

                        var nodeG = g.append("g")
                              .attr("stroke","#fff")
                              .attr("stroke-width", 1.5)
                            .selectAll("circle")
                            .data(nodes)
                            .enter().append("circle");

                        //Larger radius for nodes with larger PageRank
                        function tick(){
                              lineG.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;})
                              nodeG.attr("cx", function(d){return d.x;})
                                .attr("cy",function(d){return d.y;})
                                .attr("r", function(d){return
radiusScale(+d.PageRank);});
                        }

                        //Same color for nodes within same components
                        var colorScale = d3.scaleOrdinal(d3.schemeCategory20);
```

```
                        nodeG.style("fill",function(d){
                              console.log(d.Component)
                              if (d.Component == 1){
                                    return "#000000";
                              }
                              else {
                                    return "#DAA520";
                              }


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

The plot is as below: