INTRODUCTION TO WEB SCIENCES:
Assignment 7

Babitha Bokka

8 November 2014

Contents

1 Question 1:

Create a graph of karate club before and after the split using D3.

1.1 Approach

Step :1 The initial step taken is writing a program(initialJSON.py) to get a JSON file but the output(graph.json) of the program didn't have the weights of the graph.

Step :2 So, I started with karate.GraphML. This was the file used in assignment 6. I wrote a program generateJSON.py which uses BeautifulSoup to parse the XML file and get a JSON file format output.

Step :3 This output file is given as input to graph.html which generates a visualization of karate club graph before split.

Step :4 I was really confused and was writing many programs to split the graph and show the change in my visualization but nothing worked out well to show the transition after split. I an took advice from Mr. Victor he suggested me to highlight the nodes.

Step 5: My visualization don't split them a part but when mouse is clicked on any node the two split groups are highlighted.

1.2 Description of generateJSON.py

1. Program uses Beautiful soup
2. Finds all the nodes and data.
3. Gets the respective attributes.
4. Writes to a file.
5. BeautifulSoup makes life easier if you understand how to work with it.

1.3 Description of graph.html

1. The program needs a JSON input file which converts it to visualization.
2. The two events in the program are mouseover and mousedown.
3. Mouseover highlights the nodes when the mouse is over or clicked.
4. Mousedown gets back to the normal graph.

1.4 Source Code

1.4.1 generateJSON.py

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

import sys
from bs4 import BeautifulSoup

#Main Function
def main():
    cou =33
    htmlfile = open('karate.GraphML', 'r')
    datafile = open('karateData.json', 'w')
    soup     = BeautifulSoup(htmlfile)
    #print soup.prettify()
    datafile.write('{\n "directed": false,\n "graph": [ \n  [ \n  "name",\n  "Zachary\'s karate Club" \n  ]\n ],\n')
    # start of nodes
    datafile.write(' "nodes": [\n' )
    for node in soup.find_all('node'):
        node_attrs = dict(node.attrs)
        id_number  = node_attrs[u'id']
        data_faction, data_name   = node.find_all('data')

        data_faction_value = data_faction.contents
        data_name_value    = data_name.contents

        id_number_split = id_number.split('n');
        count           = id_number_split[1]
        # write to a a file
        datafile.write('  {\n')
        datafile.write('   "id": ')
        datafile.write( id_number_split[1])
        datafile.write(',\n')

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

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

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

     # end of nodes
#--------------------------------------------------------------------------------------------------
    # start of links
    datafile.write('\n "links": [\n' )
    for edge in soup.find_all('edge'):
        edge_attrs  = dict(edge.attrs)
```

```
        edge_source = edge_attrs[u'source']
        edge_target = edge_attrs[u'target']

        edge_source_split = edge_source.split('n')
        edge_target_split = edge_target.split('n')

        data_weight       = edge.find('data')
        data_weight_value = data_weight.contents

        datafile.write('  {\n')
        datafile.write('   "source": ')
        datafile.write(edge_source_split[1])
        datafile.write(',\n')

        datafile.write('   "target": ')
        datafile.write(edge_target_split[1])
        datafile.write(',\n')

        datafile.write('   "weight": "')
        datafile.write(data_weight_value[0])
        datafile.write('"\n ')

        #print edge_source_split[1]
        if edge_source_split[1] == 32:
            datafile.write(' }\n ],\n "multigraph": false\n } ')
        else:
             datafile.write(' },\n')

    # end of links

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(1)
```

1.4.2 graph.html

```
<!DOCTYPE html>
<meta charset="utf-8">

<html>
<head>
    <h1 align="center">
      Karate Club D3 visualization
    </h1>
    <style>

    .node {
      stroke: #fff;
      stroke-width: 1.5px;
    }

    .link {
      stroke: #999;
      stroke-opacity: .6;
    }

    </style>

    <script src="d3.min.js"></script>
</head>

<body>

<script>


var width = 960,
    height = 600;

var color = function( x ){
    if(x == 1 ){
        return "#FF0000";
    }
    else{
        return "#0000FF";
    }
}

var force = d3.layout.force()
    .charge(-200)
    .linkDistance(100)
    .size([width, height]);

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

d3.json("dataa.json", function(error, graph) {
  force
      .nodes(graph.nodes)
      .links(graph.links)
```

```
      .start();

  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("circle")
      .attr("class", "node")
      .attr("r", 5)
      .style('fill', function(d) { return color(d.color); })
      // To handle the mouse click events
      .on("mouseover", mouseover)
      .on("mousedown", mousedown)
      .call(force.drag);

  var nodelabels = svg.selectAll(".nodelabel")
       .data(graph.nodes)
       .enter()
       .append("text")
       .attr({"x":function(d){return d.x;},
              "y":function(d){return d.y;},
              "class":"nodelabel",
              "stroke":"black"})
       .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("cx", function(d) { return d.x; })
        .attr("cy", function(d) { return d.y; });

    // To label the nodes
    nodelabels.attr("x", function(d) { return d.x; })
              .attr("y", function(d) { return d.y; });

  });

  });
 // Functions to handle the mouse events
  function mouseover(d)
  {
    d3.selectAll('.node').style('fill', function(d) { return color(d.faction); });
  }
  function mousedown(d)
  {
    d3.selectAll('.node').style('fill', function(d) { return color(d.color); });
  }

</script>
```

```
113 </body>
114 </html>
```

1.5 Input

1.5.1 karate.GraphML

```
<?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. Journal of Anthropological Research Vol. 33, No. 4
    452-473</data>
    <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>
  <edge source="n0" target="n1">
      <data key="weight">4</data>
    </edge>
    <edge source="n0" target="n2">
      <data key="weight">5</data>
    </edge>
  <edge source="n0" target="n3">
      <data key="weight">3</data>
    </edge>
  </graph>
</graphml>
```

1.6 Output Files

1.6.1 karateData.json


```
{
 "directed": false,
 "graph": [
  [
  "name",
  "Zachary's karate Club"
  ]
 ],
 "nodes": [
  {
   "id": 0,
   "faction": 1,
   "name": "Mr Hi"
  },
  {
   "id": 1,
   "faction": 1,
   "name": "Actor 2"
  }
 ],
 "links": [
  {
   "source": 0,
   "target": 1,
   "weight": "4"
  },
  {
   "source": 0,
   "target": 2,
   "weight": "5"
  }
 ],
 "multigraph": false
}
```

1.6.2 Visulaization

1. Figure 1 represents the group before split.
2. Figure 2 represents a transition after split by highlighting the nodes.

```
To view the visualization please go the link.
http://www.cs.odu.edu/~bbokka/cs594/graph.html
```

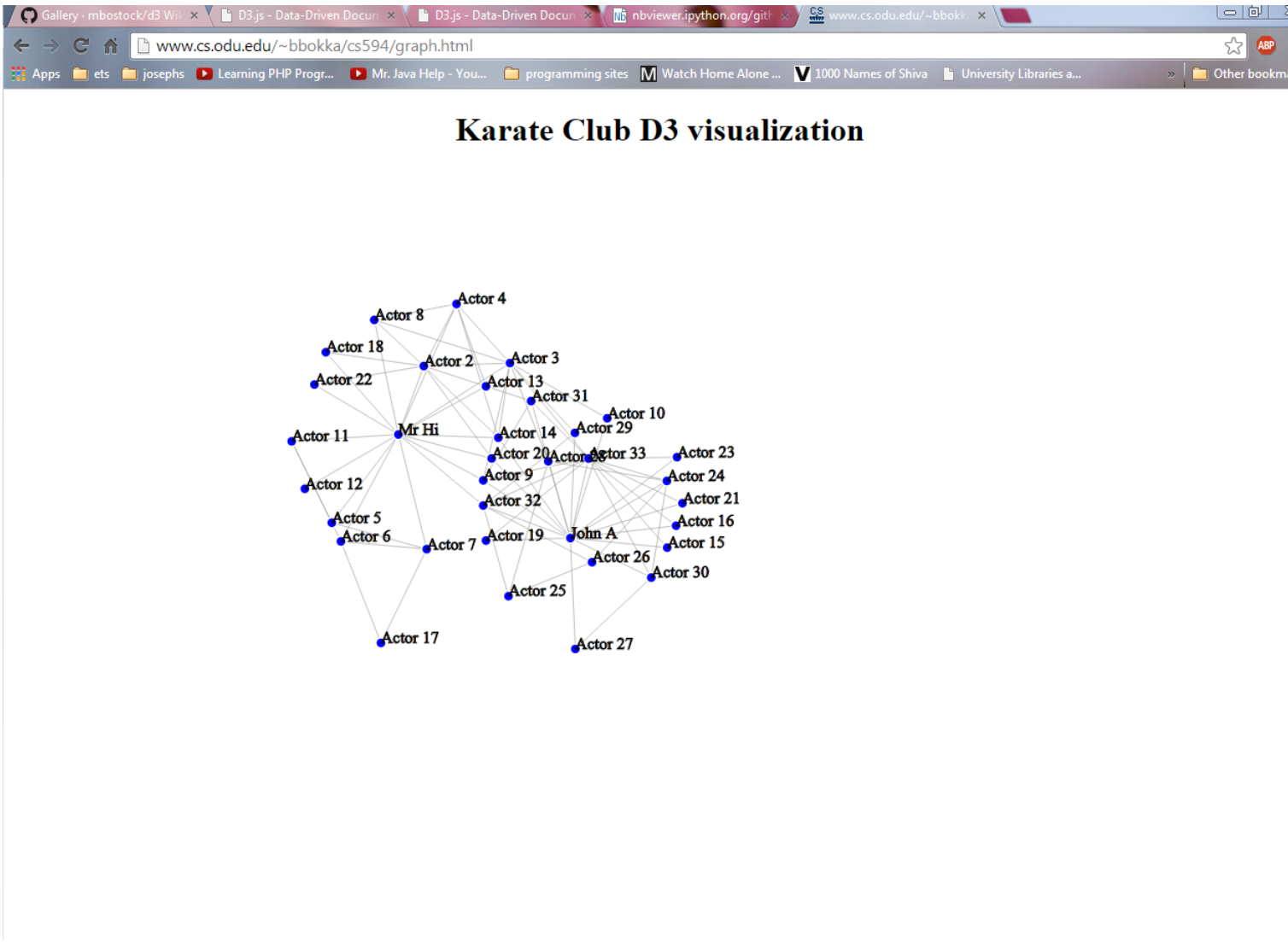

Figure 1: screen shot of the karate club

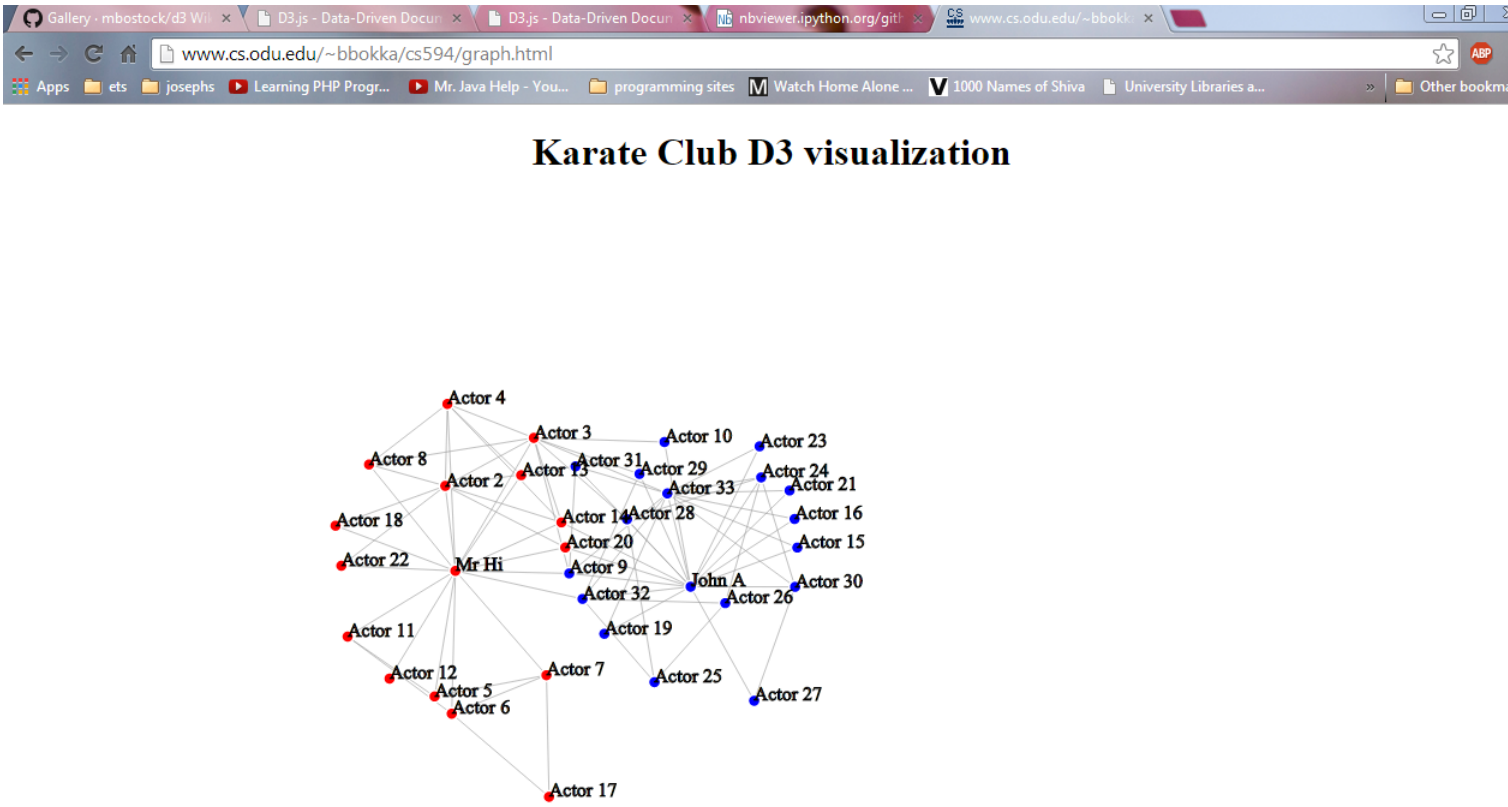

Figure 2: screen shot highlighting the two groups

References

[1] Stack overflow for xml parsing. http://stackoverflow.com/questions/4071696/python-beautifulsoup-xml-parsing.

[2] Template for d3. http://nbviewer.ipython.org/github/rossant/euroscipy2014/blob/master/04_dataviz.ipynb

[]