

INTRODUCTION TO WEB SCIENCES: Assignment 7

Babitha Bokka

11 November 2014

Contents

- 1 Question 1: 2**
 - 1.1 Approach 2
 - 1.2 Description of generateJSON.py 2
 - 1.3 Description of graph.html 2
 - 1.4 Source Code 3
 - 1.4.1 generateJSON.py 3
 - 1.4.2 graph.html 5
 - 1.5 Input 8
 - 1.5.1 karate.GraphML 8
 - 1.6 Output Files 9
 - 1.6.1 karateData.json 9
 - 1.6.2 Visulaization 10

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.

1.4.1 generateJSON.py

46 #

```

51     edge_source = edge_attrs[u'source']
52     edge_target = edge_attrs[u'target']
53
54     edge_source_split = edge_source.split('\n')
55     edge_target_split = edge_target.split('\n')
56
57     data_weight = edge.find('data')
58     data_weight_value = data_weight.contents
59
60     datafile.write(' {\n')
61     datafile.write('     "source": ')
62     datafile.write(edge_source_split[1])
63     datafile.write(',\n')
64
65     datafile.write('     "target": ')
66     datafile.write(edge_target_split[1])
67     datafile.write(',\n')
68
69     datafile.write('     "weight": ')
70     datafile.write(data_weight_value[0])
71     datafile.write('"\n')
72
73     #print edge_source_split[1]
74     if edge_source_split[1] == 32:
75         datafile.write(' }\n ],\n "multigraph": false\n } ')
76     else:
77         datafile.write(' },\n')
78
79 # end of links
80
81 if __name__ == "__main__":
82     try:
83         main()
84     except KeyboardInterrupt:
85         sys.exit(1)

```

1.4.2 graph.html

```
1 <!DOCTYPE html>
2 <meta charset="utf-8">
3
4 <html>
5 <head>
6   <h1 align="center">
7     Karate Club D3 visualization
8   </h1>
9   <style>
10
11     .node {
12       stroke: #fff;
13       stroke-width: 1.5px;
14     }
15
16     .link {
17       stroke: #999;
18       stroke-opacity: .6;
19     }
20
21   </style>
22
23   <script src="d3.min.js"></script>
24 </head>
25
26 <body>
27
28 <script>
29
30
31 var width = 960,
32     height = 600;
33
34 var color = function( x ){
35   if( x == 1 ){
36     return "#FF0000";
37   }
38   else{
39     return "#0000FF";
40   }
41 }
42
43 var force = d3.layout.force()
44   .charge(-200)
45   .linkDistance(100)
46   .size([width, height]);
47
48 var svg = d3.select("body").append("svg")
49   .attr("width", width)
50   .attr("height", height);
51
52 d3.json("dataa.json", function(error, graph) {
53   force
54     .nodes(graph.nodes)
55     .links(graph.links)
```

```

56     .start();
57
58     var link = svg.selectAll(".link")
59         .data(graph.links)
60         .enter().append("line")
61         .attr("class", "link")
62         .style("stroke-width", function(d) { return Math.sqrt(d.value); });
63
64     var node = svg.selectAll(".node")
65         .data(graph.nodes)
66         .enter().append("circle")
67         .attr("class", "node")
68         .attr("r", 5)
69         .style('fill', function(d) { return color(d.color); })
70         // To handle the mouse click events
71         .on("mouseover", mouseover)
72         .on("mousedown", mousedown)
73         .call(force.drag);
74
75     var nodelabels = svg.selectAll(".nodelabel")
76         .data(graph.nodes)
77         .enter()
78         .append("text")
79         .attr({ "x": function(d){return d.x;},
80               "y": function(d){return d.y;},
81               "class": "nodelabel",
82               "stroke": "black" })
83         .text(function(d){return d.name;});
84
85     force.on("tick", function() {
86         link.attr("x1", function(d) { return d.source.x; })
87             .attr("y1", function(d) { return d.source.y; })
88             .attr("x2", function(d) { return d.target.x; })
89             .attr("y2", function(d) { return d.target.y; });
90
91         node.attr("cx", function(d) { return d.x; })
92             .attr("cy", function(d) { return d.y; });
93
94         // To label the nodes
95         nodelabels.attr("x", function(d) { return d.x; })
96                     .attr("y", function(d) { return d.y; });
97     });
98
99
100 });
101 // Functions to handle the mouse events
102 function mouseover(d)
103 {
104     d3.selectAll('.node').style('fill', function(d) { return color(d.faction); });
105 }
106 function mousedown(d)
107 {
108     d3.selectAll('.node').style('fill', function(d) { return color(d.color); });
109 }
110
111 </script>
112

```

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


1.5 Input

1.5.1 karate.GraphML

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
5         http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6 <!-- Created by igraph -->
7   <key id="name" for="graph" attr.name="name" attr.type="string"/>
8   <key id="Citation" for="graph" attr.name="Citation" attr.type="string"/>
9   <key id="Author" for="graph" attr.name="Author" attr.type="string"/>
10  <key id="Faction" for="node" attr.name="Faction" attr.type="double"/>
11  <key id="name" for="node" attr.name="name" attr.type="string"/>
12  <key id="weight" for="edge" attr.name="weight" attr.type="double"/>
13  <graph id="G" edgedefault="undirected">
14    <data key="name">Zachary&apos;s karate club network</data>
15    <data key="Citation">Wayne W. Zachary. An Information Flow Model for Conflict and
16    Fission in Small Groups. Journal of Anthropological Research Vol. 33, No. 4
17    452-473</data>
18    <data key="Author">Wayne W. Zachary</data>
19    <node id="n0">
20      <data key="Faction">1</data>
21      <data key="name">Mr Hi</data>
22    </node>
23    <node id="n1">
24      <data key="Faction">1</data>
25      <data key="name">Actor 2</data>
26    </node>
27    <node id="n2">
28      <data key="Faction">1</data>
29      <data key="name">Actor 3</data>
30    </node>
31    <edge source="n0" target="n1">
32      <data key="weight">4</data>
33    </edge>
34    <edge source="n0" target="n2">
35      <data key="weight">5</data>
36    </edge>
37    <edge source="n0" target="n3">
38      <data key="weight">3</data>
39    </edge>
40  </graph>
41 </graphml>
```

1.6 Output Files

1.6.1 karateData.json

```
1 {
2   "directed": false ,
3   "graph": [
4     [
5       "name",
6       "Zachary's karate Club"
7     ]
8   ],
9   "nodes": [
10    {
11      "id": 0,
12      "faction": 1,
13      "name": "Mr Hi"
14    },
15    {
16      "id": 1,
17      "faction": 1,
18      "name": "Actor 2"
19    }
20  ],
21  "links": [
22    {
23      "source": 0,
24      "target": 1,
25      "weight": "4"
26    },
27    {
28      "source": 0,
29      "target": 2,
30      "weight": "5"
31    }
32  ],
33  "multigraph": false
34 }
```

1.6.2 Visulaization

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

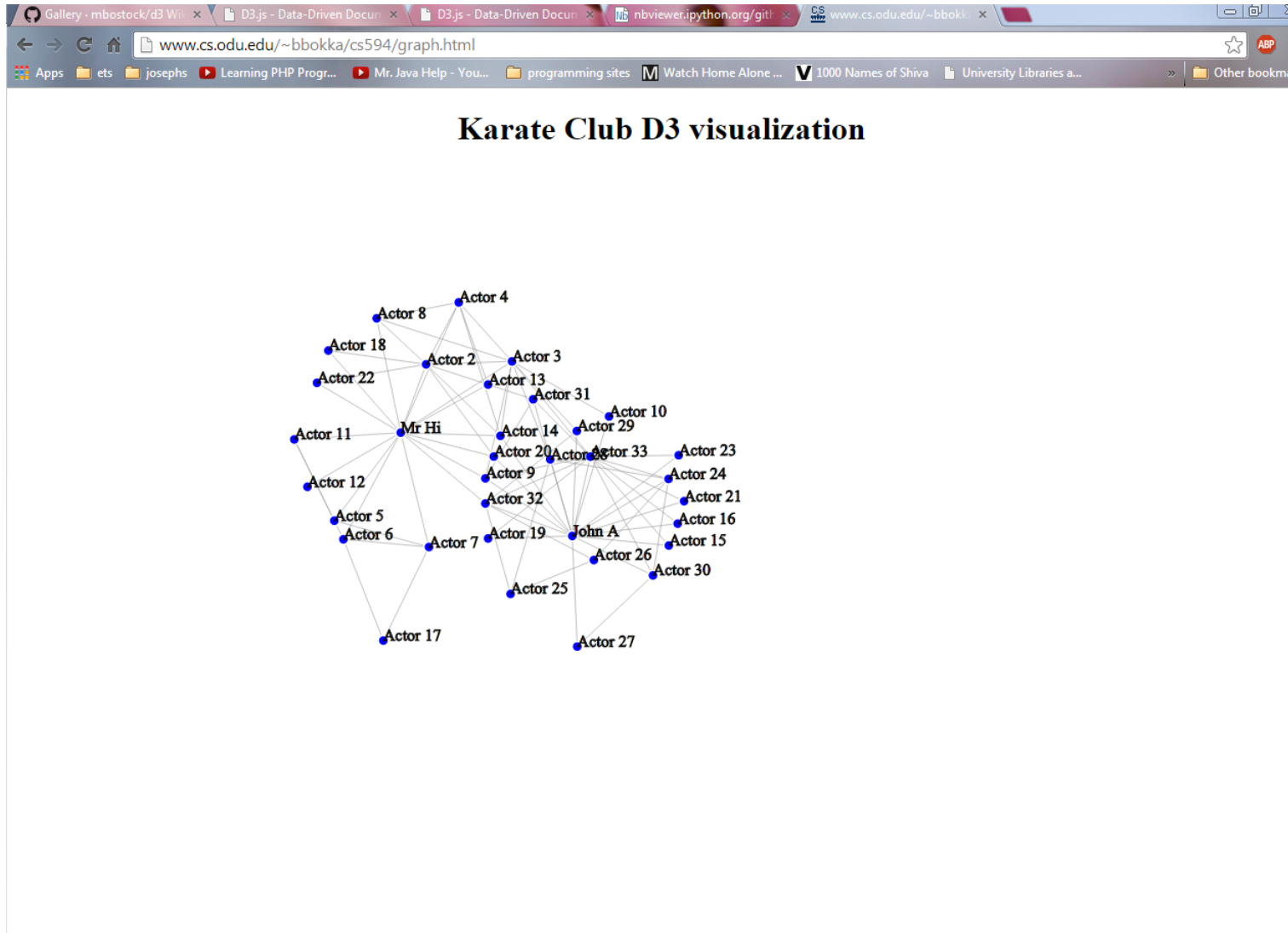


Figure 1: screen shot of the karate club

Karate Club D3 visualization

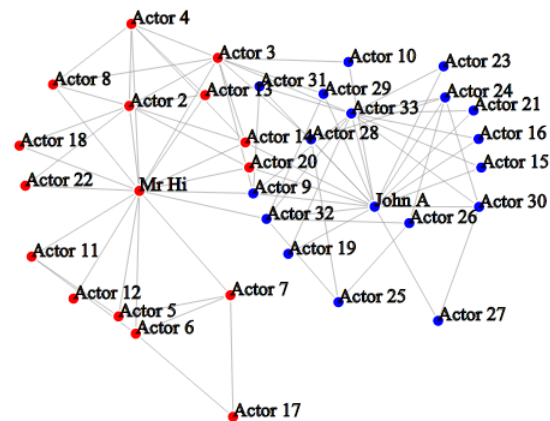


Figure 2: screen shot highlighting the two groups

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

□