Introduction to Web Science: Assignment #7

Dr. Nelson

Alexander Nwala

Thursday, November 6, 2014

Contents

Problem 1

Using D3, create a graph of the Karate club before and after the split.

- Weight the edges with the data from:
http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/zachary.dat

- Have the transition from before/after the split occur on a mouse click.

SOLUTION 1

With the use of D3.js (a JavaScript library for manipulating documents based on data) [2], and the Force-Directed Graph template for undirected graphs (http://bl.ocks.org/mbostock/4062045), the following solution was achieved.

The solution for this problem is outlined by the following steps:

1. **Convert the Karate club graph to json:** With the use of BeautifulSoup [1], as outlined in Listing 1, the Karate club graph, karate.GraphML was converted to karateClubGraph.json.

Listing 1: Convert karate.GraphML to karateClubGraph.json

```
#convert Karate club GraphML file to a json file
if( len(parentNodes) > 0 ):

     for i in range(0, len(parentNodes)):
          factionAndNodeName = parentNodes[i].findAll('data')
          faction = factionAndNodeName[0].text
          nodeName = factionAndNodeName[1].text

          stringToWrite = '\t {"name": "' + nodeName + '", "faction": ' + faction
              + ', "color": 1 },' + '\n'
          #remove comma
          if( i == len(parentNodes)-1 ):
               stringToWrite = '\t {"name": "' + nodeName + '", "faction": ' +
                   faction + ', "color": 1 }' + '\n'

          outputFile.write(stringToWrite)


     outputFile.write('\t],' + '\n')
     outputFile.write('\t"links":' + '\n')
     outputFile.write('\t[' + '\n')

parentEdges = soup.findAll('edge')
if( len(parentEdges) > 0 ):
     for i in range(0, len(parentEdges)):

          edgeWeight = parentEdges[i].find('data')
          #print edgeWeight.text

          #data = parentEdges[i].find('edge')
```

```
            sourceTargetDate = str(parentEdges[i])

            indexOfStart = sourceTargetDate.find('source="')
            indexOfEnd = sourceTargetDate.find('>')
            sourceTargetDate = sourceTargetDate[indexOfStart:indexOfEnd]

            sourceTargetDate = sourceTargetDate.split('"')
            sourceData = sourceTargetDate[1][1:]
            targetData = sourceTargetDate[3][1:]

            stringToWrite = '\t {"source": ' + sourceData + ', "target": ' +
                targetData + ', "weight": ' + edgeWeight.text + ', "id": "e' + str(i
                +1) + '" },\n'

            if( i == len(parentEdges)-1 ):
                 stringToWrite = '\t {"source": ' + sourceData + ', "target": ' +
                     targetData + ', "weight": ' + edgeWeight.text + ', "id": "e' +
                     str(i+1) + '" }\n'
...
```

2. **Color code graph based on before/after split:** In karateClubGraph.json, every node was given the same "color" attribute, but a different color based on it's "faction" attibute as outlined below

```
...
 {"name": "Mr Hi", "faction": 1, "color": 1 },
 {"name": "2", "faction": 1, "color": 1 },
 {"name": "3", "faction": 1, "color": 1 },
 {"name": "4", "faction": 1, "color": 1 },
 {"name": "5", "faction": 1, "color": 1 },
 {"name": "6", "faction": 1, "color": 1 },
 {"name": "7", "faction": 1, "color": 1 },
 {"name": "8", "faction": 1, "color": 1 },
 {"name": "9", "faction": 2, "color": 1 },
 {"name": "10", "faction": 2, "color": 1 },
 ...
```

3. **Toggle node color on click:** As shown in Listing 2. All nodes have been wired to the on click event. This means when any node is clicked, the function specialNodeClick, is called. And this function simply toggles the fill color of the node from the same color to the faction of the node.

Listing 2: Toggle node color

```
//OnClick event block
var node = svg.selectAll(".node")
          .data(graph.nodes)
          .enter()
          .append("circle")
          .attr("class", "node")
          .on("click", specialNodeClick)
          .attr("id", function(d) { return d.name; })
          .attr("r", 5)
```

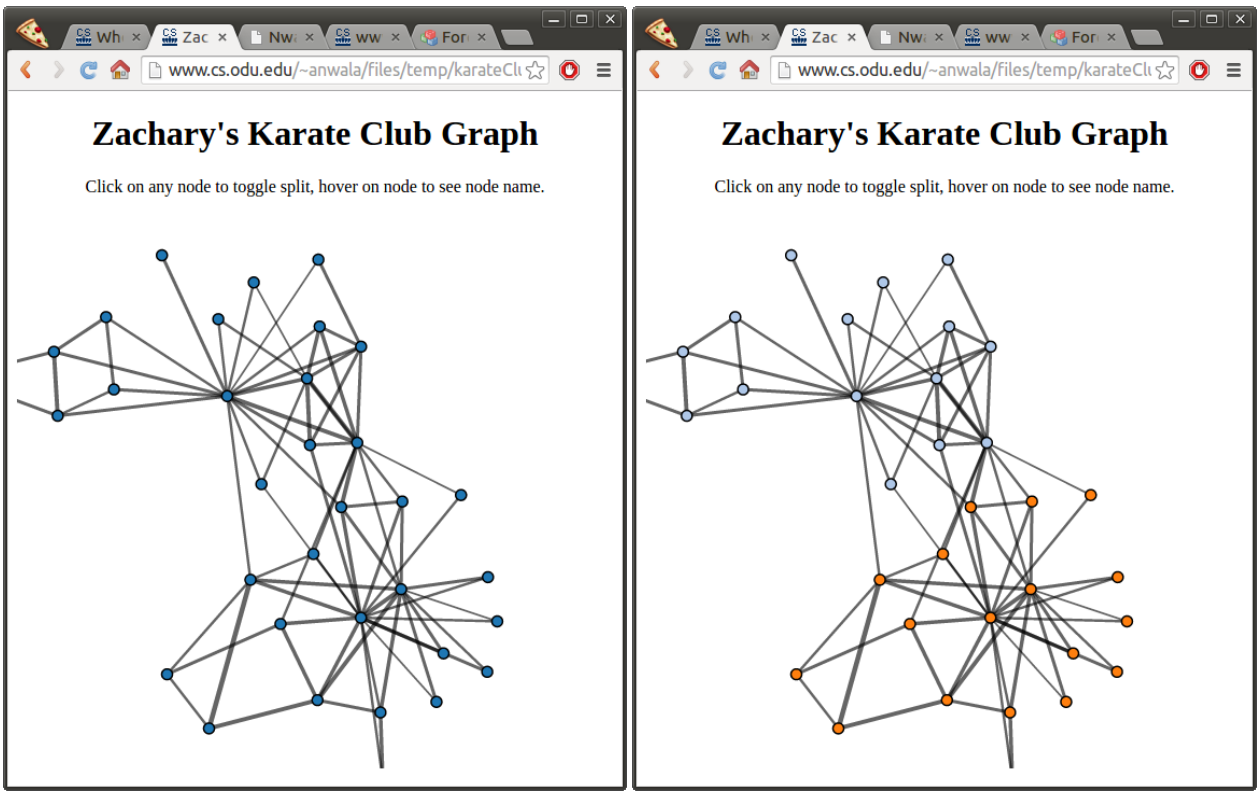
```
            .style("fill", function(d) { return color(d.color); })
            .call(force.drag);

function specialNodeClick(d)
{
  if(globalClickFlag==1)
  {
    d3.selectAll('.node').style('fill', function(d) { return color(d.faction - 10)
        ; });
    globalClickFlag = 0;
  }
  else
  {
    d3.selectAll('.node').style('fill', function(d) { return color(d.color); });
    globalClickFlag = 1;
  }
}
```

CONCLUSION 1

The result the following operations (Figure 1) can be seen at http://www.cs.odu.edu/~anwala/files/temp/karateClub.html

Figure 1: Karate Club Graph



(a) Before Split

(b) After Split

Problem 2

Use D3 to create a who-follows-whom graph of your Twitter account. Use my twitter account ("@phonedude_mln") if you do not have an interesting number of followers. Attractiveness of the graph counts!

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

SOLUTION 2

With the use of D3.js (a JavaScript library for manipulating documents based on data) [2], and the Force-Directed Graph template for directed graphs (http://bl.ocks.org/mbostock/1153292), following solution was achieved.

The solution for this problem is outlined by the following steps:

1. **Develop an algorithm to get the followers of followers:** In order to get the followers of followers across a variable degree, I derived the iterative algorithm (A1) presented in **algorithm1.txt**
2. **Convert A1'S output to a links file:** This was achieved by a python script as shown in Listing 3.

CONCLUSION 2

The result the following operations (Figure 2) can be seen at http://www.cs.odu.edu/~anwala/files/temp/whosFollowingWho.html

References

[1] BeautifulSoup. http://www.crummy.com/software/BeautifulSoup/bs4/doc/. Accessed: 2014-11-06.

[2] Data Driven Documents. http://d3js.org/. Accessed: 2014-11-06.

Figure 2: Who follows who

(a) Who follows who