Project 2: **Social Network**
Comp 475 – Web Science
100 Points

This project is composed of four components:

1. Writing a Python script to create a social network by converting data found on the Web into GraphML format.
2. Writing a Python script to read the GraphML file, perform a characterization of the network, and apply the Girvan-Newman algorithm to the network.
3. Create a visualization of the network using NodeXL, Graphviz, Gelphi, D3.js, or any other visualization software of your choosing.
4. Present your work in class in a 10 minute presentation.

Step 1: Creating a Social Network from Web Data

You need to locate a data source on the Web in order to create a social network. Here are some ideas:

1. Actors starring in a movie
   http://www.imdb.com/interfaces

2. Biblical names
   http://moreno.ss.uci.edu/data.html#bible
   http://www-958.ibm.com/software/data/cognos/manyeyes/visualizations/co-occurrences-of-names-in-the-new-3

3. Twitter users discussing a particular topic
   https://twitter.com/search?q=%23marchMadness&src=typd

4. Teams competing against each other
   http://apps.washingtonpost.com/sports/apps/live-updating-mens-ncaa-basketball-bracket/bracket/2013/
   http://www.databasefootball.com/

You should save the web pages or text files to disk manually and write your Python script to parse through the saved files and create a GraphML file. If you are parsing through HTML, you will likely want to write some regular expressions. Structured text files will likely be easier to parse.

Step 2: Network Characterization and Girvan-Newman Algorithm

In this step you will write another Python script which will read the GraphML file you created with the script from the previous step. You will use the NetworkX Python library to do this. http://networkx.github.com/

Download and install NetworkX on your own machine. Then you can read in your GraphML file like so:

```
import networkx as nx
G = nx.read_graphml("mygraph.graphml")
```

Below is an example file that represents the graph on the right.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
```

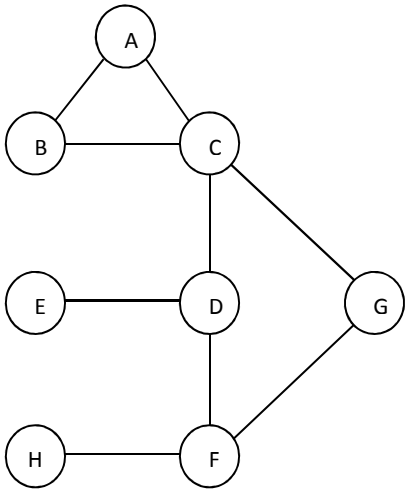
```
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">

<graph id="G" edgedefault="undirected">

<node id="A"></node>
<node id="B"></node>
<node id="C"></node>
<node id="D"></node>
<node id="E"></node>
<node id="F"></node>
<node id="G"></node>
<node id="H"></node>

<edge source="A" target="B"></edge>
<edge source="A" target="C"></edge>
<edge source="B" target="C"></edge>
<edge source="C" target="D"></edge>
<edge source="E" target="D"></edge>
<edge source="D" target="F"></edge>
<edge source="F" target="H"></edge>
<edge source="F" target="G"></edge>
<edge source="C" target="G"></edge>

</graph>
</graphml>
```



Your Python script must report the following information about your network:

1. Number of nodes.
2. Number of edges.
3. Number of connected components.
4. The diameter (longest shortest path).
5. The five nodes with the highest clustering coefficients.
6. The five nodes with highest betweenness centrality (node betweenness).
7. The edges that would be removed using the Girvan-Newman algorithm when splitting the given graph with *N* components into *N*+5 components.

If the graph above were used as input, this is the output your program would produce:

```
Nodes: 8
Edges: 9
Components: 1
Diameter: 4

Top 5 clustering coefficients
1. A 1.0
2. B 1.0
3. C 0.16666666666666666
4. D 0.0
5. E 0.0

Top 5 betweenness centrality
1. C 11.0
2. D 9.0
3. F 7.0
4. G 3.0
5. E 0.0

Girvan-Newman algorithm
Components: 1
```

```
Delete edge ('D', 'C') with betweenness 10.0
Components: 1
Delete edge ('F', 'G') with betweenness 16.0
Components: 2
Delete edge ('D', 'F') with betweenness 4.0
Components: 3
Delete edge ('G', 'C') with betweenness 3.0
Components: 4
Delete edge ('D', 'E') with betweenness 1.0
Components: 5
Delete edge ('B', 'C') with betweenness 1.0
Components: 5
Delete edge ('A', 'B') with betweenness 2.0
Components: 6
```

Please format your output to match my example.

There are NetworkX functions to compute all the information above. You only need to write some custom code to display the information in the correct format. You'll need to search the NetworkX documentation for the functions that perform the needed operations, and they should not be difficult to find.

For the Girvan-Newman algorithm, you will not need to write any code to compute edge betweenness because NetworkX already has a function to compute this. You will only need to write a loop which removes one edge at a time by finding the edge with the largest edge betweenness and deleting it. If more than one edge has the same max betweenness values, just pick one of the edges to delete.

Step 3: Create a Visualization

You can use any network visualization tool you'd like, but you need to create a single image (or more if you'd like) that gives a good picture of what your graph looks like. You should introduce color. The best thing to color will likely be tightly clustered groups of nodes. When you are presenting your visualization to the class, you will want to explain what you did to produce your image using the tool you have chosen. Those who use multiple techniques and produce more fascinating images will be given higher grades in this category.

Step 4: Present Your Work

Each of you will be given 10 minutes to present your work. Presentations will be given two class periods after your project is due. You may use PowerPoint slides if you think it would be beneficial, but it is not necessary. Your presentation should contain the following:

1. Show us the website from which you obtained your data and explain you chose this data. Discuss the social network you wanted to create, what the nodes and edges represent.

2. Show us the GraphML that your Python script produced after parsing the web data.

3. Show us your visualization. If you use a tool other than NodeXL, it likely will not be installed on the podium computer, so you should include screenshots of it in a PowerPoint presentation to illustrate what you did to produce your image.

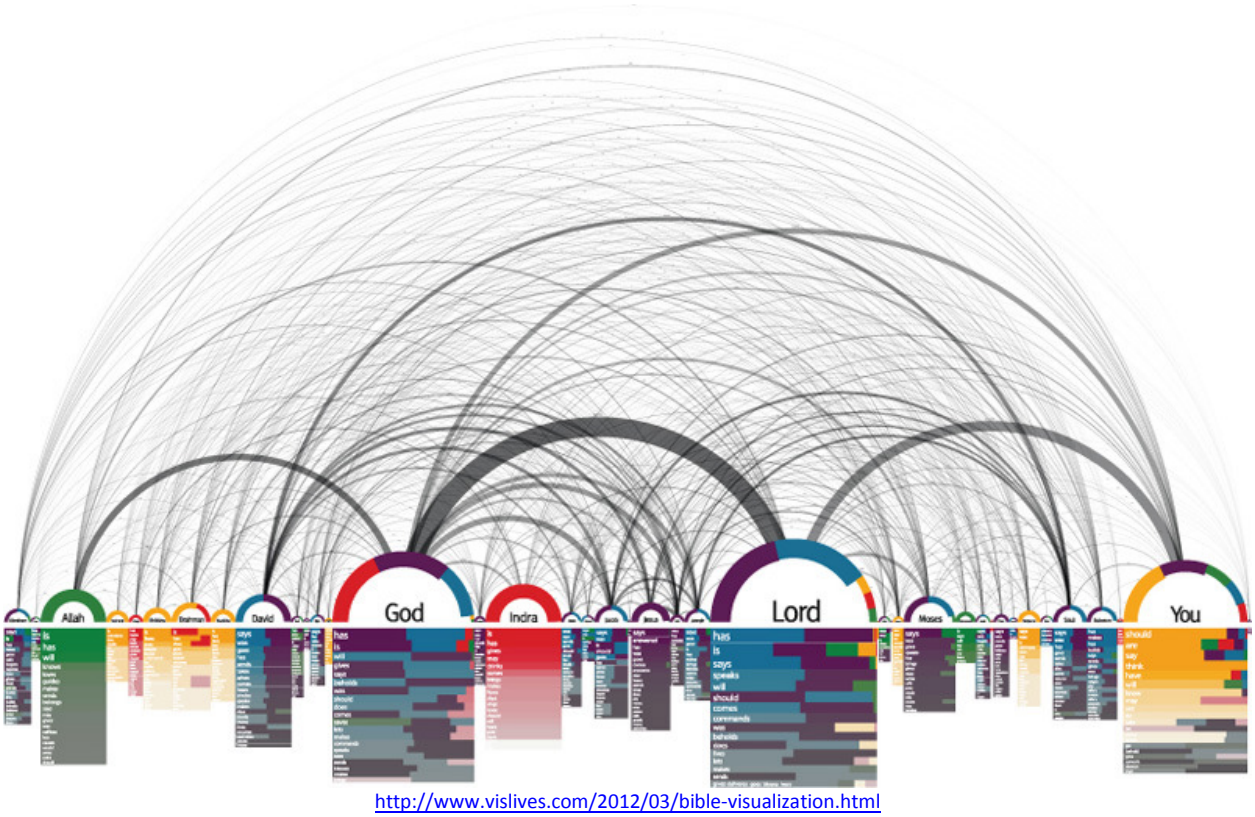
Grading

Your project will be evaluated like so:

1. 20% based on your scripts ability to transform web data into a GraphML file.
2. 45% based on your script properly showing the network characteristics and Girvan-Newman algorithm.
3. 20% based on the quality of your visualization. I will be comparing your visualization to everyone else's to determine this score.
4. 15% based on your presentation. A presentation that is organized, communicates the required information effectively, and is close to 10 minutes will receive full credit.

Turn In

Submit a zip file containing your Python source code, your raw data files and GraphML file, and your visualization (.png) to Easel before the due date. I will be replacing your own GraphML file with my own to test your script's characterization values, so make sure it is easy for me to modify your code to read in a different file.



http://www.vislives.com/2012/03/bible-visualization.html