EECS 428 / ECE 578
DATA VISUALIZATION
Spring 2015

ASSIGNMENT 4

Due Date: *Sunday, June 7th, 2015, 23:59*

(16 Points)

Assignment Submission: *Turn in your assignment by the due date through LMS. Prepare and upload* ***one zip file****. Name the zip file as <your first name>_<your last name>_ assignment1. See the question for what you should return.*

You can discuss JavaScript, D3 and question with each other. ***However, implementation must be your own; you must neither copy from nor provide assistance to anybody else (including online resources). If you need guidance for any question, talk to the instructor or TA.***

Visual Aid for Plagiarism Detection

In this assignment, you will develop an interactive graph-based visual to help us detect plagiarism cases in the assignments. We have a software tool to measure the similarity of the assignments. When the tool is ran on a pair of assignments submitted by students (assignments A and B), it calculates A's similarity to B and B's similarity to A and create an HTML file that includes these similarity scores (and a lot more). Note that A's similarity to B and B's similarity to A can be different. For example, A can be completely similar (100%) to B (if A is fully copied from B), but B's similarity to A can be lower if, for example, the owner of B have not shared the solution of the most-valued question and kept it to himself. A sample data set has been provided to you to use in this assignment. Open up some of the HTML files and get familiar with their format.

Your goal in this assignment is to represent the similarities in a graph form as show in reference sketch in Figure 1 and help us to identify the offenders. Each node in the graph will represent a student (i.e., an assignment) and the weight of an edge between two nodes is the similarity of the assignments of the corresponding students. Graph will be an undirected graph. *For the weight of the edges, you will use 'NormalizedMetric' value provided in the input data.* For an edge between A & B, the '*NormalizedMetric*' value will be set to the average of the similarity of A to B and the similarity of B to A in this assignment.

We pre-processed the HTML files in the data set and created a JSON file for you (assignments.json). Each object in JSON file has 5 fields:

1- Student_A : The owner of the first assignment in the assignment pair that we check
2- Student_B: The owner of the second assignment in the assignment pair that we check
3- A_B_Similarity: Similarity of the assignment of student_A to the assignment of student_B.
4- B_A_Similarity: Similarity of the assignment of student_B to the assignment of student_A.

Node Score Calculation:

Method: [✓ Sum] Max

Only Use Edge Scores Over:

0% 100%

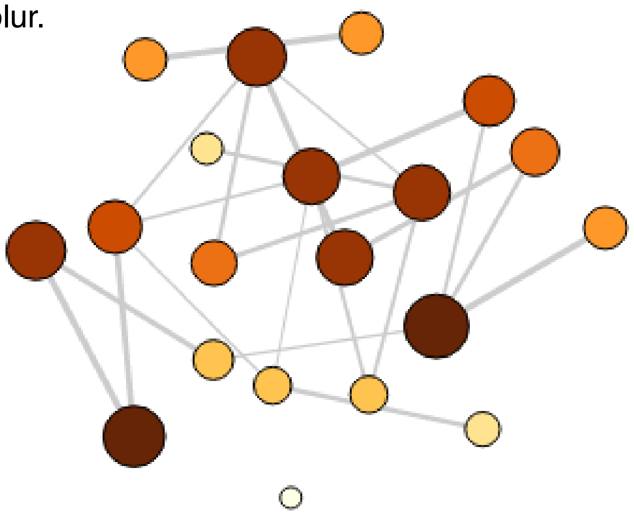
Visual:

Show Only the Edges with Scores Over:

0% 100%

Node Labels: [✓ Hide] Show

Skoru hesaplamak için node'lardan değil de edge'lerden for döndürüp, bağlı node'lara skorları eklemek daha hızlı olur.



Rank 1 : Student_0 -- score: 17

Rank 2 : Student_11 -- score: 17

Rank 3 : Student_15 -- score: 14

Rank 4 : Student_13 -- score: 14

Rank 5 : Student_16 -- score: 13

Rank 6 : Student_4 -- score: 13

Rank 7 : Student_8 -- score: 13

Rank 8 : Student_17 -- score: 12

Rank 9 : Student_19 -- score: 12

Rank 10 : Student_14 -- score: 12

...

Figure 1: Reference Sketch

5- NormalizedMetric: An aggregate value derived from A_B_Similarity and B_A_Similarity. In this assignment, it is set to the average of A_B_Similarity and B_A_Similarity. **Note that you will use this value as the weight of the edge between A & B in your graph.**
6- File: The name of the actual file (an HTML file) that has the assignment similarity information for this pair of assignments. You will need this file name to open the file as part of this assignment as discussed later.

<u>**Force-Directed Graph**</u>
Use the information in the JSON file to draw a force-directed graph. Each node in the graph represents a student (assignment). Edges are ***undirected*** and represent the similarity between corresponding assignments. You will use the *NormalizedMetric* value as an aggregate similarity score for each edge as discussed above.

<u>**Node Score Calculation**</u>
Once you create the graph model, calculate a score for each node (student). Provide 2 different options for node score calculation. Use Radio buttons to select one of these 2 options.

1- **Sum:** Sum up the weight (*NormalizedMetric* value) of all edges connected to a node and use it as the node score
2- **Max:** Look at the weight (*NormalizedMetric* values) of all edges connected to a node and select the maximum of them as the node score

Edge filtering during Node score calculation:
In case of **sum** calculation, to further customize the node score calculation, provide a slider bar to select the minimum edge weight to use in the calculation. Slider range should be from 0% to 100% with 1% increments. Based on the selected value on the slider, if the edge weight is less than the selected value by the slider, ignore it in the edge calculation. **Not that this feature is only applicable when calculating the sum.**

<u>**Color and Size of Nodes & Edges**</u>
<u>Nodes:</u>
Size: The radius (or area) of nodes should be proportional to the calculated node scores. Since the variation can be very high across the nodes, make sure to select minimum & maximum sizes for radius as in Assignment 3 and map the node score to this range by using a scale. You can select min and max numbers based on your graph area. Node should not be too small (i.e. difficult to see and select) or too large (i.e. covers a very large part of the graph).
Color: Node colors should be used to highlight the variation among node scores. Ideally, use a color spectrum from white to red, representing nodes from lower to higher scores (see some example color spectrums)

<u>Edges:</u>

Thickness: Thickness of the edges should be proportional to the corresponding edge weights. Similar to nodes, make sure to select minimum & maximum thickness levels to create visually appealing graphs.

Color: Use grey color for all edges

Visual Customization

Edge filtering: Only show the edges with an edge weight (NormalizedMetric value) that is larger than a user selected limit. Provide a slider bar for the user to select the minimum edge score limit. Slider range should be from 0% to 100% with 1% increments. Based on the selected value on the slider, if the edge weight is less than the selected value, do not show it on the graph. **Note that even if a node is under the selected threshold and not shown in the graph, it can be still used in node score calculation.**

Node Labels:

Provide an option to show the corresponding student's name on the top of each node. Provide radio buttons to show or hide the labels.

Events on Edges & Nodes

Edges:

Mouseover: Highlight the edge and the connected nodes (let's say the nodes are A and B). Additionally, a tooltip should show the source/target node names (A and B), the edge weight (NormalizedMetric), A_B_Similarity and B_A_Similarity.

Click: When clicked an edge, show, ***in a new browser window,*** the HTML file that has the corresponding assignment similarity information. ***Note that the HTML file name is provided in JSON file. So, simply open the corresponding HTML file in a new browser window.***

Files: Assume that JSON and HTML files are under a directory called 'input'. This directory is located at the same level as your index.html file.

Nodes:

Mouseover: A tooltip shows the node score and the corresponding user name.

Click: Highlight the node & the edges and all other nodes connected to this node. To highlight nodes, set node stroke color (the boundary of the circle) to black (or any other color you prefer) and increase stroke thickness. When the node is clicked again or another node is clicked, the highlights are removed. If another node was clicked, this new node and its neighbors are highlighted after removing the previous highlights.

When a node is clicked, the corresponding student in the ranked student/node list should be highlighted as well as discussed below.

Ranked Student (Node) List:

At the bottom of the graph, there should be an ordered list (highest to lowest) of all students based on the corresponding node scores. Each line in the list should have the rank, student name and the score information at minimum. Note that when the node scores change (switch between sum and max or the edge filtering limit change for node score calculation by using slider), the student list should updated and re-sorted

Click Event:

When a node is clicked (node click event), in addition to other actions discussed before, the corresponding student in the ranked list should be highlighted as well.

Please return in a directory:

All your HTML, JavaScript and CSS files (you can have sub-directories if you prefer). Your main HTML file should be named as ***index.html***. When we open *index.html* in our web browser (*Chrome* will be used for grading), the visuals should show up in the main page. All input data we provide (JSON and HTML files) will be under a directory called ‘input’. This directory will be located at the same level as your index.html file.