CS 5001 – Applied Social Network Analysis
Spring 2021
HW #1

20 points

Submit to Canvas by 11:59 p.m. on Friday, Feb. 5th

What to Do:

For this assignment you are to write a **Python** program that will **retrieve pages from Wikipedia and compare them using graph analysis.** The methods for doing this have been covered in lecture; you just have to put them together and tweak them a bit.

We want to compare the results of **Wikipedia** searches on **"Toototabon"** and **"Rabbits (film)**" (weird stuff!). This will be **2 separate searches** (**NOT one search** on both terms). Limit **each** of these searches to **3 "layers" deep** in terms of traversing their links. As in the example done in lecture, cut down each graph you build to only include **nodes that have degree ≥ 2 and no self-loops**.

For **each** of the 2 graphs that you produce (i.e., one for the **"Toototabon"** search and one for the **"Rabbits (film)**" search), output **a list of the subjects in decreasing order of in-degree**.

Next, for **each** of the 2 graphs that you produce (i.e., one for the **"Toototabon"** search and one for the **"Rabbits (film)**" search), compute **Similarity Rank** for the nodes in that graph. **Output the pairs of nodes in each graph that have similarity ≥ 0.015**; so, you'll have a list of similar nodes for the first graph and a list of similar nodes for the second graph. Don't just output the indexes of the nodes, **output their labels! Also output the similarity rank value for each of those pairs of nodes in each graph.** If we weren't triaging our searches, we would cross-compare similar nodes between the 2 graphs. But there probably would be few matches between graphs in this assignment. **NOTE: The similarity rank computation could take several minutes to run!**

Now we want to see what is **different** between the 2 graphs. To do this using **networkx.difference** the nodes must be exactly the same in the 2 graphs, which they likely are not! You will have to write code to **remove nodes that are in the "Toototabon" graph and not in the "Rabbits (film)", and vice versa**[1]. After computing the difference, you probably want to **remove nodes that have degree 0**; they're probably not very interesting. Use **networkx.info** to **report the main statistics about the difference graph**. The graph is probably too big to display it legibly; if you can find a way to do so, you can also include a screenshot showing what it looks like.

Note that the 2 graphs in this assignment are too big to compare using **edit distance**; if you try that (you should use networkx.optimize_graph_edit_distance), you'll find that it takes hours (days?) to get a result. So, this is **not required** for this assignment.

We do, however, also want to see what is the **same** in the 2 graphs. If these were sets, we would use **intersection**. To do networkx.difference, you had to make 2 graphs that had the same nodes. Now maybe you want to find the intersection of the edges of those graphs; Python has a set intersection method. Alternatively, **networkx has an intersection function**. You can do it either way. After finding the intersection, you probably want to **remove nodes that have degree 0**; again, they're probably not as interesting. Use **networkx.info** to **report the main statistics about the "intersection" graph**. If you can find a way to display the graph legibly, include a screenshot showing what it looks like.

What to Turn In:

You need to submit (via Canvas) the following (all as a **single** pdf file):

(1) A listing of **your source code**.

(2) A screen shot showing the **number of nodes, number of edges, average in-degree, and average out-degree** for each of the two graphs after cutting them down to only include **nodes that have degree ≥ 2 and no self-loops.**

(3) A screen shot showing **a list of the subjects sorted by in-degree** for each of the two graphs after cutting them down to only include **nodes that have degree ≥ 2 and no self-loops.**

(4) The results of determining the **similar nodes for each of the 2 graphs**.

[1] See https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms.operators.binary.difference.html for a suggestion on how to do this.

(5) A screen shot showing the results (networkx.info statistics) of finding **the difference between the 2 graphs**. Inclusion of a screenshot showing what the difference graph looks like is optional; don't include it unless you can make it readable. Including a picture of a big blob would be like giving the grader a Rorschach test!

(6) A screen shot showing the results (networkx.info statistics) of finding **the "intersection" between the 2 graphs**. Inclusion of a screenshot showing what this graph looks like is optional; don't include it unless you can make it readable. Again, a big blob is meaningless and can cause night terrors.

Grading:

Here's how many points each task is worth:

Task	Points Possible
Build 1st graph 3 layers deep, remove loops, remove nodes with degree < 2, output stats	**2**
Display subjects in 1st graph sorted by in-degree	**2**
Find similar nodes in 1st graph	**3**
Build 2nd graph 3 layers deep, remove loops, remove nodes with degree < 2, output stats	**2**
Display subjects in 2nd graph sorted by in-degree	**2**
Find similar nodes in 2nd graph	**3**
Find difference between graphs, output stats	**4**
Find intersection between graphs, ouput stats	**2**
Total	**20**