CS 5001 – Applied Social Network Analysis
Spring 2021
HW #5

30 points

Submit to Canvas by 11:59 p.m. on Monday, Mar. 29, 2021

What to Do:

For this assignment you are to write code that will **analyze a social network of Marvel Universe (comic book) characters.** This assignment is a bit contrived/convoluted in that it has you doing tasks in **Python, Neo4j,** and even **Gephi**! However, the objective is to get you to practice several things that we have discussed recently in class (e.g., Gephi, Neo4j, the Python Neo4j API, and blockmodeling).

Posted on Canvas is a file[1] (**hero-network.csv**). The very first row in the file contains the "headers" for the data: **Name1,Name2**. Each subsequent row contains the names of two Marvel Universe characters; because a character's name can contain blanks (and commas), each name has double quotes around it. Names are separated on a line with a comma.

Each name will correspond to a node in our social network; we want an (undirected) edge between the 2 names that are specified on a line in the data file (these are Marvel Universe characters who interacted with each other at some time). There are 6426 nodes and 167219 edges; the average degree is ~52.

You are to do all of the following (take careful note of what is to be done in **Neo4j**, what is to be done in **Python**, and what is to be done in **Gephi**):

(1) **Create an undirected graph** in **Python**, reading in the data from hero-network.csv. Output the number of nodes, number of edges, and average degree to make sure those numbers agree with the statistics that are stated above.

[1] This file was created on a Windows machine. If you're using it on a Unix/Linux machine, you may want to run *dos2unix* on it before using it to remove any weird characters like '\r'

(2) In **Python** use the **Louvain method to partition the nodes into communities**. Output the number of communities found and the modularity of the partitioning.

(3) From **Python**, perform **blockmodelling** on the graph based on the Louvain partitioning from step (2). Output the number of nodes, number of edges, and average degree in the blocked graph.

(4) From **Python**, for each **community**, find the **node with the highest degree**; if there is a tie, use whatever method you want to pick one of the nodes with highest degree. **Change the name of each "supernode" in the blocked graph** to be the name of the node with the highest degree in that community plus "_GROUP". Also, change any spaces in the node name to underscores and remove any commas in the name (those could be problematic for what we're going to do in a subsequent step). For example, suppose you have a community in which "IRON MAN" is the node with highest degree. After blockmodelling, there will be a supernode for that community; its label will be an integer. Change that node's (integer) label to the string "IRON_MAN_GROUP".

(5) From **Python**, connect to Neo4j and **run Neo4j queries that create a new graph** which contains the nodes and edges of the blockmodelled graph. Note: You may need to start Neo4j on your computer before establishing the connection (i.e., before creating the 'driver' object) via the Python API. To show that this worked correctly, **include a screenshot of this graph from the Neo4j browser interface.**

Hint: It might be easiest to create csv file(s) of your graph data from Python, copy those files to the proper Neo4j import directory (you'll have to figure out where that is!), and then make your Neo4j queries be ones that **LOAD CSV…** and **CREATE…** nodes and edges from the csv file data.

(6) From inside of **Neo4j**, **run Louvain partitioning on the graph you created** (yes, this is the graph that we already partitioned into communities in Python). Your query should output the name of each node and the community it is assigned to, grouped by community; in your homework submission include a screenshot of your query to do the Louvain partitioning as well as the output.

(7) From **Gephi**, load a file containing **the blocked graph edge data** created in step (4). Run Louvain partitioning on it and then **assign colors to the nodes according to the Louvain partitions**. Make a screenshot of this graph that includes the node labels. Feel free to adjust other features to make it pretty!

(8) Answer the following question (in writing): what knowledge/usefulness do we gain by **performing community partitioning a second time** on this social network? Your answer must be specific (and relevant) to this particular dataset which involves crime-fighting superheroes. Maybe your answer could include a use-case for how this second partitioning could be helpful to the Agents of S.H.I.E.L.D. (google that if you don't know what it is).

What to Turn In:

Here's what you need to submit via Canvas (all as a **single** pdf file):

(1) A listing of **your source code**. **BE SURE TO COMMENT YOUR CODE!!!**

(2) A screen shot showing the **output** for all of the things you were asked to do, **in exactly the same order** you were asked to do them!

(3) A written answer for the very last task.

Grading:

Here's how many points each task is worth:

Task	Points Possible
Create graph in Python, output stats	2
Do Louvain partitioning in Python, output stats	2
Do blockmodelling in Python, output stats	3
Relabel supernodes in Python	4
Run Neo4j queries in Python to create graph database	10
Run Louvain partitioning in Neo4j	3
Run Louvain partitioning in Gephi	3
Analyze usefulness of double partitioning	3
Total	**30**