CS7280: Network Science

[illegible]IGNMENT 3

Note:

- In this assignm[illegible]o submit the two jupyter notebooks (**Centrality-Ass[illegible]ity-Detection-Assignment**) with all the plots, values and comments that have been asked in the assignment as well as code.
- Please also submit **requirements.txt** so that we may be able to replicate your Python dependencies to run your code as needed.
- With Anaconda, you can do this by running: conda list -e > requirements.txt
- Please submit **unzipped** two ipynb files, requirements.txt file. Part 3 can be put in a markdown box in the ipynb file.

Use the following networks given in the Assignment Packet

1. Yeast transcription network (2002) (**Directed, Unweighted**): Network of operons and their pairwise interactions, via transcription factor-based regulation, within the yeast Saccharomyces cerevisiae
2. US airport networks (2010) (**Undirected Weighted**): Network of flights among the 500 busiest commercial airports in the United States, in 2002. Weights represent the number of seats available on the flights between a pair of airports.

Part - 1: Centrality Metrics (30 Points)

For the first part of the assignment, use the previous two network datasets for the computation of various centrality metrics. For the Yeast transcription network, although it's directed, we import as undirected network for this assignment. We recommend the use of the NetworkX library available with python.

For each of the two network datasets:

- Compute and print the top-10 nodes according to the centrality metrics listed below.
- Compare and contrast the ranking of the top-10 nodes obtained from the different centrality metrics using Jaccard Similarity Index heatmaps. For grading consistency, please use the following order of metrics in the rows and columns of the heatmaps.

Centrality Metrics

1. Eigen-vector Centrality

2. Katz Centrality (make sure the parameters lead to converged result)
3. Pagerank Centrality (make sure that the parameters lead to converged result)
4. Closeness Centrality (consider both networks as unweighted for this metric)
5. Harmonic Centrality (consider both networks as unweighted for this metric)
6. Shorte[illegible]Centrality (consider both networks as unweighted for this metric)

Helpful Hints

- Use the functi[illegible]py instead of *katz_centrality* in case the latter throws any error.
- Please make su[illegible]on signatures from NetworkX documentation and pass the correct "weight" parameter.
- Note that Jaccard Similarity is defined on sets, so ordering is not important. Please see https://en.wikipedia.org/wiki/Jaccard_index
- Note that NetworkX requires alpha parameters for Katz centrality, which is reciprocal of lambda we discussed in the lecture. **PLEASE BE CAREFUL IN SETTING IT**. You may want to compare your results with Eigenvector centrality as a sanity check. Since these metrics are closely related, getting quite different results may indicate a problem with convergence.
- By default, the closeness centrality function of networkX performs a modified version of the computation presented in the lesson. Therefore, you need to pass *wf_improved=False* parameter to use the desired version.

Part - 2: Community Detection (65 Points)

Use the following community detection algorithms:

- C-Finder (i.e K-Clique) (Available in NetworkX)
- Greedy Modularity Maximization algorithm: (Available in NetworkX)
- Louvain Method for community detection : You may use *nx.louvain_communities()* if your network x version is 2.7+.

A. (25 points) Run the three community detection algorithms on Zachary's Karate Club dataset available through the NetworkX library.
   a. Tune the parameters of the algorithms to obtain the best possible results based on your knowledge of the algorithm, for example use critical density threshold for K-clique (see Lesson-7). Produce a plot for the communities detected by each of the three algorithms. Use different colors for different communities. In case one node belongs to more than one community, give it a different color and label it.
   It is likely that you won't obtain the exact ground-truth, as given in lecture, but please attempt to closely match that answer. Along with the plots of the communities, report the value of the parameter K in the C-finder algorithm. Calculating density of the graph may help in determining the value of K.

Hint : do not use the number of communities knowledge from the ground truth available, the intent of the question is to understand the effect of various hyper-parameters and algorithms on the communities detected.

b. With th[...] is likely that some nodes will belong to more than one commu[...] understanding of the C-Finder algorithm, comment on that ca[...]

B. (25 points) Use [...]work generator in the second jupyter notebook of the assignment pa[...]

a. First consider the value of *mu = 0.1* and compute the corresponding graph. Run the three community detection algorithms listed above, and plot the corresponding communities.

Tip: If the communities are not visible due to the size of the network, try not coloring nodes outside communities, plotting with *spring_layout,* and increasing the size of the figure

b. Consider the two algorithms, modularity maximization and Louvain, increment the value of mu from 0.1 to 1 in 10 steps and obtain the communities using each of the two algorithms. Then, compute the normalized mutual information (NMI) metric comparing the accuracy of each algorithm to the ground-truth. Plot the NMI values as a function of mu for the two algorithms in the same plot. You will need to implement the NMI function at the given space of the jupyter notebook and call whenever required. Present the results and the plot for 10 iterations for each of the mu values. Make sure you implement the NMI function manually using the math library. Use of other libraries is not permitted.The

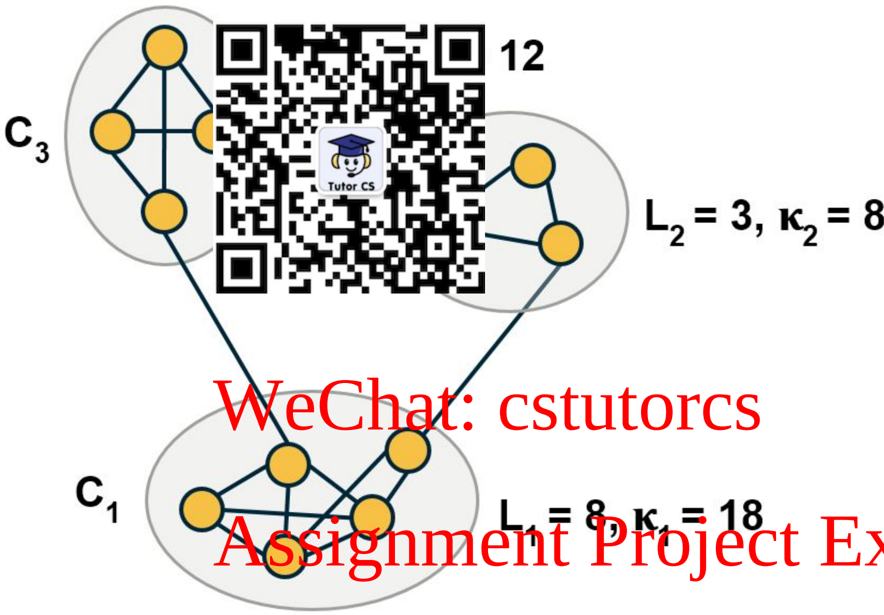
C. (15 points) In this section you will run the community detection algorithms for the networks listed at the start of the assignment.

a. Run the two community detection algorithms (modularity maximization and Louvain) for the two networks listed above. Plot the community size distributions obtained through the different algorithms.

b. (**Optional**) Evaluate the two algorithms (modularity maximization and Louvain) against each other by using the Adjusted Rand Index (ARI). This metric is used to compare clustering results on the basis of similarities and dissimilarities between the clusters obtained. Report the ARI values for the pair of algorithms (you can read more about the ARI metric at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html)

Part - 3: Knowledge Question (5 Points)

Answer the following food for thought question from Lesson 7 in a markdown box you created at the end of ipynb file for Part2:



Use the modularity formula derived in "L7: Modularity Metric- Derivation" to calculate the modularity of each of the following partitions on the above network:

a) all nodes are in the same community,

b) each node is in a community by itself,

c) each community includes nodes that are not connected with each other,

d) a partition in which there are no inter-community edges.