Homework 4: PageRank Stability and Community Structure on Graphs

The goal of this homework is to study how to compute PageRank, to analyze the stability of PageRank as the graph evolves, i.e., grows or shrinks, and to analyze the community structures on graphs.

In order to prevent you spending much time on implementing PageRank and community detection, we provide you with implementations in Python and a relatively small data set about a social network on which you can run/test the implementations. The dataset contains a directed network of friendships and familylinks between users of the website hamsterster.com. There are 2426 nodes and 16631 edges in this data set. It is available from Canvas in HW4.zip. The archive contains the original and processed files. The edges.txt contains two columns, "source node id" and "target node id". The original dataset can be found at http://konect.uni-koblenz.de/networks/petster-hamster.

Your task is (1) to compute the PageRank of nodes in these data sets to explore the effect of graph evolution (i.e., removing/adding some nodes/edges randomly, proportional to the node degrees or any other characteristics/statistical properties of the network) on the stability of PageRank; and (2) to detect the community structures of the given graph and analyze the results.

Instructions

Task 1: PageRank Stability on Evolving Graphs

Mathematically, the computation of PageRank is to find the eigenvector corresponding the eigenvalue equal to 1, i.e.,

$$r = Mr$$

where M is the transition matrix (with or without teleport) and r is the PageRank vector. Thus, a direct method to compute PageRank is to calculate this principle eigenvector. In practice, *Power iteration* method is more widely used.

a) Compute the PageRank on the given graph and report your findings. You can use the provided script. The script implements PagaRank in Python and is based on a complex network library, NetworkX.

b) Compute the effects of nodes/edges removal/addition. In this part, you need to design a framework to study the behavior of PageRank on the evolving graph. Ingredients:

- a way to evolve the graph. You can decide whether starting from the full graph as it is in the dataset, you want to go back towards its origins or to grow it. We expect you to add or remove
    - only edges;
    - nodes and corresponding edges.

  You need to do this at least in two different ways:
    - uniformly at random;
    - at random but proportional to node degree or any other statistic.

- a measure to compare PageRank, e.g. the value-based error or rank-based error measures can be used. One way to do this is explained below.

Consider the following example: there are 3nodes and the rank/order and PageRank values are shown in the table. The baseline PageRank is computed on the full graph with a method that is known to give an exact unique solution. PageRank_i and Rank_i are the results generated by some other methods.

Baseline PageRank	Baseline Rank	PageRank_1	PageRank_2	PageRank_i	Rank_1	Rank_2	Rank_i
0.50	1	0.40	0.60	…	3	1	…
0.25	2	0.30	0.25	…	4	3	…
0.20	3	0.20	0.10	…	1	2	…
0.15	4	0.10	0.05	…	2	4	…
…	…	…	…	…	…	…	…

For the rank-based error, the error can be defined as

$$Error_{rank} = \sum_{i=1}^{n} \frac{|rank - rank_{baseline}|}{rank_{baseline}}$$

where rank is generated by the used method. Therefore, the rank-based error for Method 1 (PageRank_1 and Rank_1) is

$$Error_{rank} = \frac{|3-1|}{1} + \frac{|4-2|}{2} + \frac{|1-3|}{3} + \frac{|2-4|}{4} + \cdots$$

And the rank-based error for Method 2 (PageRank_2 and Rank_2) is

$$Error_{rank} = \frac{|1-1|}{1} + \frac{|3-2|}{2} + \frac{|2-3|}{3} + \frac{|4-4|}{4} + \cdots$$

Similarly, the value-based error can be defined as

$$Error_{value} = \sum_{i=1}^{n} \frac{|PageRank - PageRank_{baseline}|}{rank_{baseline}}$$

You are also encouraged to define your own metrics for the comparison.

Important notes:

- It is possible that when PageRank values change from one graph to the other, the corresponding ranking may change, but may also stay the same.
- When you remove the nodes, the above metrics may need adjustments as the new RageRank and baseline lists are of different lengths.
- Since there is randomness in the process of removing nodes/edges, it is a good idea to repeat the process several times and report the mean and standard deviation.

- When you reduce or grow the graph – you can try two ways of initializing PageRank value *r* and analyze how many iterations did it take to converge with a) a random initialization, and b) with previously computed PageRank.

Task 2: Community Structure on Graphs

Detect communities on the given graphs using some community detection algorithm. In this part, a script of community detection in Python will be provided. This script implements the Louvain algorithm (a greedy optimization method for modularity maximization) and is based on two Python libraries introduced in the lecture: NetworkX (https://networkx.github.io/) and community (http://perso.crans.org/aynaud/communities/index.html). You need to analyze the detected communities and report what you find. For example, you can use different measures to analyze the community structures, e.g., visualization, edge density between and within communities, modularity, etc. You are also encouraged to try other community detection algorithms and other toolkits (or implement methods in different programming languages).

Grading

The homework will be evaluated based on a report on PageRank values, community structures and the analysis of the results. Your grade will be based on different components:

- 20 points: PageRank value calculation and analysis.
- 50 points: Graph evolution and PageRank values comparison.
- 30 points: Community detection and analysis.
- 10 extra bonus points can be earned for using other graph evolving strategies, different community detection algorithms and additional analytics you find interesting to apply on this dataset.

Submission deadline

When: the deadline is January 18, 17:00 (extended upon your request).
How: upload your report as pdf and a zip archive including all your other files via canvas.tue.nl.
What to submit:

- A report, explaining all the computations/analysis performed and the summaries of the findings. Whenever you found anything expected or unexpected try to argument why you got what you got.
- Enclose xls/csv files with all the results (the main results/summaries you comment on should be in the report).
- Enclose all the scripts/code you wrote yourself (or borrowed from elsewhere, indicating the source) in your experiments.