Information Retrieval Lab, Session 5: Network analysis

In this final lab session, you will perform network analysis on some synthetic and real networks. I suggest that you use some existing network analysis library like `networkx` or `igraph`, but feel free to use any other existing software or program the functionality from scratch if you prefer (and have some time available). You will have to analyze several networks; in the first section we propose to reproduce well-known facts about network models that you have seen in class. This first part is understood as a warming-up exercise. In the second part, you will have to build your own network and study it with the type of tools you have seen in theory class.

1 Analyzing network models

In class you have seen three main random network models:

Erdős-Rényi model (ER model). The ER model takes two parameters: n, the number of vertices in the resulting network, and p, the probability of having an edge between any two pairs of nodes. A graph following this model is generated by connecting pairs of vertices with probability p, independently for each pair of vertices. Erdős-Rényi graphs have $\binom{n}{2}p$ edges in expectation.

Watts-Strogatz model (WS model). The WS model takes two parameters as well: n, the number of vertices in the resulting network, and p, the probability of rewiring the edges in the initial network. A graph following this model is generated by initially laying all nodes out in a circle, and connecting each node to its four closest nodes. After that, we randomly reconnect each edge with probability p.

Barabasi-Albert model (BA model). The BA model takes two parameters: n, the number of vertices in the resulting network, and m, the number of edges a "new" vertex brings to attach itself to existing nodes. A graph in this model is generated by adding new nodes according to the preferential attachment principled until the resulting graph has the desired size.

In this first section, your task is to do two of the three following alternatives:

1. Plot the clustering coefficient and the average shortest-path as a function of the parameter p of the WS model.
2. Plot the average shortest-path length as a function of the network size of the ER model.
3. Plot a histogram of the degree distribution of a BA network. What distribution does this follow? Can you describe it?

For option (1), notice that in order to include both values — average shortest path and clustering coefficient — in the same figure, the clustering coefficient and the average shortest-path values are normalized to be within the range $[0, 1]$. This is achieved by dividing the values by the value obtained at the left-most point, that is, when $p = 0$.

For option (2), you will have to experiment with appropriate values of p which may depend on the parameter n. You will notice that for large values of n your code may take too long, compute values for n that are reasonable for you. Also, make sure that you chose values for p that result (with high probability) in connected graphs. To achieve this, you can use a result from [1] stating (in the following, think of ϵ as a small positive real number):

- If $p < \frac{(1-\epsilon)\ln n}{n}$ then a graph in $G(n, p)$ will almost surely contain isolated vertices, and thus be disconnected
- If $p > \frac{(1+\epsilon)\ln n}{n}$ then a graph in $G(n, p)$ will almost surely be connected

For option (3), choose a network that is large enough so that results are what is expected from this model.

2 Analyzing and building your own network

2.1 Generate your network

To leverage the concepts that you have seen in the first part of the course (similarity for texts, etc.), we suggest that you use a collection of texts of your choice, in order to build a network. For example:

- Using Twitter messages as nodes in your network and using a text-based similarity measure to add links between them.
- Using news gathered from the web and linking them also using some kind of text-based similarity measure.
- Using recipes as the sources of text and linking recipes based on the ingredients that they use, or the cooking methods.
- Using classical texts from public domain authors e.g. Shakespeare or Jane Austen or Cervantes or all of them.
- Using abstracts from scientific papers.

You can be as creative as you like. Your resulting network should be connected, and have at least 200 nodes.

If your data comes from Twitter you can use libraries e.g. `pattern` in Python. In case you do not find an appealing dataset, you can use any of the ones available from the SNAP project, but we do prefer that you build your own in some interesting way.

2.2 Analyzing your network

Now that you have built a network, you should use the tools seen in class to gather information about it. For each network analysis metric (seen in class) that you can reasonably[1] compute, try to interpret the result based on the knowledge you have about the network and how was generated. For instance:

1. Describe the network a little. How many edges and nodes does it have? What is its diameter? And transitivity? And degree distribution? Does it look like a random network? Apply a pagerank on the nodes and, if it is possible, visualize the network with node sizes proportional to their pagerank.
2. Now, use a community detection algorithm of your choice from the list provided. How many nodes does the largest community found contain? Plot the histogram of community sizes. If possible, plot the graph with its communities.
3. Does the result make any sense, given that you know how you created the network?

3 Deliverables

To deliver: You must deliver a report (6-8 pages) describing your results. You also have to hand in the source code of your implementations.

Procedure: Submit your work through the Racò at `https://raco.fib.upc.edu/` as a single zipped file.

Deadline: Work must be delivered within **3 weeks** from the lab session you attend. Late deliveries risk being penalized or not accepted at all. If you anticipate problems with the deadline, tell me as soon as possible.

References

[1] Paul Erdős and A Rényi. On the evolution of random graphs. Publ. Math. Inst. Hungar. Acad. Sci, 5:17–61, 1960.

[1] Here, *reasonably* means that you spend time in the order of a few hours, not days, writing code and executing it on your network.