Link Prediction Lab

Social Data Management

January 14th, 2019

The goal of this lab session is to implement queries on probabilistic graphs and the greedy influence maximization algorithm, using Python and the `networkx` package, by using a Jupyter notebook.

Documentation to support this lab can be found at:

- NetworkX: `https://networkx.github.io/documentation/stable/`,
- Matplotlib: `https://matplotlib.org/`,
- Tutorial on how to use NetworkX and Matplotlib in Jupyter: `https://github.com/networkx/notebooks`.

1 Installation and First Steps

1. If not present, install Jupyter and the NetworkX and Matplotlib Python packages.
2. Download the Jupyter notebook called `lab3.ipynb` and the graph `karate`. The files must be in the same folder.
3. Run the Jupyter notebook, and check that all cells are executed correctly.

2 Link Scoring Functions

In this exercise we will implement the link scoring functions for a variety of possible scoring functions. An example implementation is given in the method `common_neighbors`, for a given graph G and a pair of nodes i and j. Ideally, all functions should have the signature `score_func(G,i,j)` so that they can be easily used in the link prediction functions (next exercise).

Implement the following link scoring functions:

1. Neighbour-based measures: Jaccard coefficient, and preferential attachment measure
2. Inverse Distance score
3. Personalized PageRank score (use a fixed value of $\alpha = 0.2$)
4. Random, in which the link is given a random score in $[0, 1]$

3 Link Prediction

Using the already implemented method `link_list`, create a method `link_prediction(G,k,i,score_func)` where the parameters are:

- `G` the input graph,
- `k` the number of links to be predicted,
- `i` the node starting from which the link needs to be predicted, and
- `score_func`, the function for link scoring (implemented in the previous exercise) as a *parameter*.

The `link_prediction` function returns the best k new links, starting from i, by their value of `score_func`.

1. Implement and test the `link_prediction` function, for a variety of `k` values and link score functions.
2. Evaluate link prediction for `karate` by the *leave-one-out* method. Proceed as follows: remove one (existing) link from the network, and try to guess it using the resulting graph and the method `link_prediction`. Repeat this process for *all* links in the graph.
3. Track the precision and recall in function of the parameters `k` and `score_func`. Precision and recall are defined as follows:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}},$$
$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}.$$