Probabilistic Graphs and Influence Maximization Lab

Social Data Management

December 17th, 2018

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 `network_analysis_lab.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 Reachability and Expected Spread

In this exercise, we will evaluate the reachability and expected spread of the nodes in the `karate` graph, by following the

1. Test the `reachability` function as implemented in the notebook, with various values for the probability and number of rounds. What do you notice?
2. Using the `reachability` function, compute the expected spread for each node in the karate graph, for varying values of the parameter p (e.g., 0.01, 0.1, 0.5). Output the 5 best nodes along with their spread, and highlight them on the original graph plot.
3. Rewrite the function `reachability` so that, instead of generating the entire graph, it does a probabilistic BFS search as described in the course. Compare the results and the running time of the two approaches (*hint*: you can use the `%%time` pre-processing instruction in the Jupyter cell you want to evaluate), both for source-target reachability and for the expected spread.

3 Influence Maximization

Implement the greedy algorithm for influence maximization, as described in the course. Proceed as follows:

1. Write a function `spread` which takes the probabilistic graph, a set of *seed nodes*, a number of rounds and outputs the estimated spread from that set.
2. Write a function `greedyIM` which take as input the probabilistic graph and a parameter k, implements the greedy influence maximization algorithm (using the `spread` function) and outputs a set containing the k best nodes and the total values of the expected spread.
3. Highlight the results on the original graph plot. Compare the results with the expected spread for k nodes chosen randomly in the graph.