Algorithmic Methods of Data Mining

Homework 4

Due: 22/12/2017, 23:59.

Instructions

You must hand in the homeworks electronically and before the due date and time.

The third homework will be performed in **groups**. If you do not know who are your group partners, email Adriano and Cristina.

Handing in:
This time, we ask you to submit your homework in a **repository in Git**. As soon as you know your group, you should fill in this form (one per group).
Be careful: the information you put in the form are the one we will use to retrieve your homework. It will be done by a bot that retrieves the repository in the status of the last commit before the deadline. For this reason, you must not make commit to the repository after the deadline. The homework repository should have as contributors all the members of the group and it should include:

1. Modules (.py) which contain the function you used.
2. README.txt (or .md) file that:
   (a) explains in details the procedures you used in both exercise 1 and 2;
   (b) clearly mentions and explains, for each module, the functions it contains.
3. A REPORT.pdf file where you extensively comment the results you obtained (including the plots you did).

For information about collaboration, and about being late check the web page.
For any questions email first Adriano and Cristina or Aris and Ioannis, if needed.

In this homework we will practice on dealing with networks. In particular, we want to carry out some information from Computer Scientists network, by applying the graph methodologies we have seen in class.

Data.
We will use the DBLP dataset. We provide you two datasets:

- full_dblp.json: json file, which needs to be parsed and contains the entire network.
- reduced_dblp.json: json file for testing and debugging, which needs to be parsed and contains a portion of the network.

1. By processing JSON file create a graph, G, whose nodes are the authors. Two nodes are connected if they share, at least, one publication. Each edge is weighted in the following way:

$$w(a_1, a_2) = 1 - J(p_1, p_2),$$

where a_1, a_2 are authors, p_1 and p_2 are the set of publication of the two authors and, $J(p_1, p_2)$ represents the jaccard similarity between these two sets of publications.

2. Here we want to compute some statistics and visualizations. Write a Python software that

   (a) given a conference in input, return the subgraph induced by the set of authors who published at the input conference at least once. Once you have the graph, compute some centralities measures (degree, closeness, betweeness) and plot them.

   (b) given in input an author and an integer d, get the subgraph induced by the nodes that have hop distance (i.e., number of edges) at most equal to d with the input author. Then, visualize the graph.

3. Here we will compute some generalized version of the Erdös number.

   (a) Write a Python software that takes in input an author (id) and returns the weight of the shortest path that connects the input author with Aris. Here, as a measure of distance you use the weight $w(a_1,a_2)$ defined previously.

   (b) Write a Python software that takes in input a subset of nodes (cardinality smaller than 21) and returns, for each node of the graph, its GroupNumber, defined as follow:

$$\text{GroupNumber}(v) = \min_{u \in I} \{\text{ShortestPath}(v,u)\},$$

   where v is a node in the graph and I is the set of input nodes.

   You must implement from scratch the algorithm to compute the shortest path. Moreover, if you can, try to think if there is a way to exploit the properties of the shortest path algorithm to avoid performing the same computations multiple times.