Social Network Analysis for Computer Scientists Fall 2018 — Assignment 2

`http://liacs.leidenuniv.nl/~takesfw/SNACS`

Deadline: October 29, 2018

This document contains 3 exercises that each consist of various numbered questions that together form Assignment 2 of the Social Network Analysis for Computer Scientists course taught at Leiden University.

For each question, the number of points awarded for a 100% correct answer is listed between parentheses. In total, you can obtain 100 points and 15 bonus points. Your grade is computed by dividing your number of points by 10. Please do not be late with handing in your work. You have to hand in the solutions to these exercises individually. Discussing the harder questions with fellow students is allowed, but writing down identical solutions is not. Hand in your solutions via the Dropbox-link on the course website, in a PDF-file, generated using LaTeX.

For each question, clearly describe how you obtained your answer. Write down any nontrivial assumptions that you make. For the exercises that require some programming, you can use any programming language, scripting language or toolkit. All practical exercises can be done on the student workstations. In any case, always clearly describe which toolkit or programming language you used and how you obtained your answer using these tools. Include relevant source code (for example, in an Appendix).

Questions or remarks? Contact the lecturer or assistant via e-mail or ask your questions during one of the weekly lectures or lab sessions. Good luck!

Exercise 1: Clustering Coefficient (20p)

The average graph clustering coefficient $C(G)$ of a connected undirected graph $G = (V, E)$ indicates the extent to which nodes cluster together, and can be defined as

$$C(G) = \frac{1}{n} \cdot \sum_{v \in V} \frac{2 \cdot |\{(u,w) \in E : (u,v) \in E \wedge (v,w) \in E\}|}{deg(v) * (deg(v) - 1)}$$

Here, $deg(v)$ is the degree of a node $v \in V$, so the number of edges adjacent to node v. We use $n = |V|$ for the number of nodes and $m = |E|$ for the number of undirected edges.

(6p) Question 1 Name a) two types of graphs that have a clustering coefficient of 0 by definition and b) a type of graph that has a clustering coefficient of 1 by definition

(4p) Question 2 Consider all possible connected undirected graphs with $n = 9$ nodes and $m = 15$ edges. Draw such a graph with a minimum average graph clustering coefficient.

(10p) Question 3 Give an algorithm for finding a connected undirected graph with a minimal average graph clustering coefficient for any given m and n. What can you say about the complexity of your algorithm?

Exercise 2: Diameter Computation (10p)

Apply the BoundingDiameters algorithm on paper to find the exact diameter (maximum distance, length of a longest shortest path) of the undirected graph in Figure 1. The algorithm is discussed during the lectures and explained in:

F.W. Takes and W.A. Kosters, Determining the Diameter of Small World Networks, in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1191-1196, 2011.
doi: `http://dx.doi.org/10.1145/2063576.2063748` or see
`http://liacs.leidenuniv.nl/~takesfw/pdf/diameter.pdf`.

You do not have to do the prepruning step discussed in the paper. Explain your steps in detail, and mention any nontrivial assumptions. As a selection strategy, alternate between choosing the node with the largest upper bound value and the node with the smallest lower bound value, breaking ties by taking the node with the highest degree. How many iterations did it take you to compute the diameter? Compare this value to the naive method for computing the diameter.

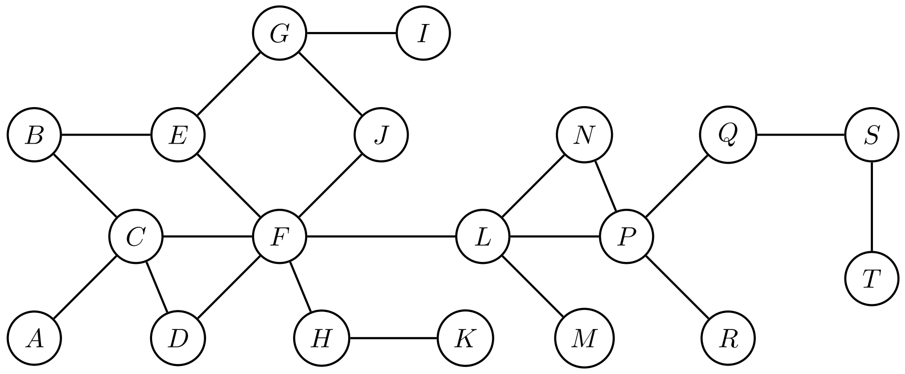

Figure 1: An undirected graph with 19 nodes and 22 edges.

Exercise 3: Twitter network analysis (70p+15p)

This is a practical exercise, for which you can use any toolkit or programming language. A Twitter dataset can be found in the Leiden ULCN environment in the shared UNIX folder `/vol/share/groups/liacs/scratch/SNACS/`
Relevant files: `twitter-small.tsv`, `twitter-larger.tsv` and `twitter.tsv`. For convenience and speed, you can copy the files to the local hard disk of your workstation, usually that is `/scratch/`.

Up until now, we have only looked at social network data which was already in a nicely formatted edge list. In practical network analysis research, this rarely ever happens. Therefore we will now work with real raw data from a Twitter crawl [2]. The dataset `twitter.tsv` contains over 450, 000, 000 tweets, crawled from June 2009 to December 2009. The file `twitter-small.tsv` contains a small subset of these tweets that can be handled with Gephi, whereas `twitter-large.tsv` contains a bit larger subset that can be analyzed using NetworkX or Graph-Tool. Each line of these files contains one tweet, consisting of three tab-separated (‘`\t`’) fields denoting the timestamp, user who sent the tweet and the content of the tweet. For example:

```
2009-07-05 14:07:18    aeneas    Hi @achilles, how are you? #old
```

In the tweet content, a word starting with the `@` symbol (such as `@achilles`) means that user `achilles` is being mentioned by user `aeneas`, indicating that the tweet by `aeneas` was directed at or specifically about `achilles`. We refer to this as “a mention”. Mentions are the most direct indication of public communication on Twitter. Tweets can also be directed at more than one user.

The *mention graph* is a Twitter network represented by a directed graph $G = (V, E)$. In this graph, the set of nodes V consists of users (anyone sending out a tweet or being mentioned by someone else in a tweet). The set of links E consists of all user pairs (x, y) such that user x mentioned user y at least once. Optionally, this network can be a weighted directed graph in case the number of times a user x mentions another user y is used for link weight. Also, the network could be analyzed over time by assigning a timestamp to each link, indicating when x first mentioned y. For the `twitter-small.tsv` dataset, answer:

(20p) Question 3.1 Extract the mention graph from the Twitter data. Relevant steps to do this could be:

- Parse the input file line by line (for example using Python or Perl)
- Generate the adjacency list: for each user (identified by its username), keep a list of the users that this user mentions, and possibly also count the number of mentions and keep track of the timestamp at which the user first mentioned the other user.
- Output the adjacency list as an edge list `csv`-file (with columns Source and Target) that you can import into Gephi or NetworkX, possibly also including columns for the Weight and Timestamp of each edge.

Discuss the steps that you took, and describe the issues that you ran into while parsing this "real-world" data, and how you solved them. For example, discuss possible text mining and parsing issues.
From your answer, it should be possible to unambiguously reproduce your results.

(12p) Question 3.2 Present relevant statistics of your mention graph, including at least the number of nodes, number of edges, density, degree distribution and (approximated) distance distribution. What can you say about the size of the giant component?

(8p) Question 3.3 Determine the top 20 users based on three different centrality measures (for example, betweenness centrality, closeness centrality and degree centrality). Discuss the results. Can you come up with a way to numerically compare the similarity of the rankings?

(8p) Question 3.4 Apply a community detection algorithm (such as Modularity in Gephi or `https://github.com/taynaud/python-louvain` in Python) to the giant component of your mention graph, and try to manually interpret and discuss the results. What is the effect of the resolution parameter?

(10p) Question 3.5 Visualize the giant component of the network (for example using Gephi), making the color of a node dependent on the community and the size of a node dependent on some sensible centrality measure. Optionally, you can try to come up with a way to incorporate the timestamp in the visualization. You could use node labels for the Twitter usernames and edge width to visualize the link weight.

(12p) Question 3.6 Run your code on the larger dataset given in the file `twitter-larger.tsv`, and answer Question 3.2 and 3.3, for example using NetworkX or Graph-Tool.

(15p, bonus) Question 3.7 Run your code on the full dataset `twitter.tsv`, and answer Question 3.2 for the giant component. This is very challenging, and may require you to systematically filter certain users and links that are not part of the giant component, for example based on some threshold for the number of mentions. If you succeed on x% of the data, you can get up to x% of the 15 bonus points. Only do this after you have finished answering all the other questions.

[2] J. Yang and J. Leskovec, Temporal variation in online media, in Proceedings of WSDM, pp. 177–186, 2011.
Available at `dx.doi.org/10.1145/1935826.1935863`