TSKS33 Hands-On Session 1

Fall 2023

Version of this document: October 17, 2023

Preparation Work (to be Done Before Coming to the Lab)

- Study the Course Material for Lectures 1 and 2.
- Familiarize yourself with the Gephi software. Here are some tutorials,

  `https://gephi.org/users`.

- Also have a look at the Stanford Network Analysis Project (SNAP) project,

  `http://snap.stanford.edu`.

  In particular, familiarize yourself with the documentation of the SNAP Python packages, and have a look at the webpages with the SNAP datasets.

- The SciPy sparse array package,

  `https://docs.scipy.org/doc/scipy/reference/sparse.html`,

  is used to represent sparse matrices in Python, and can be helpful.

Task 1: Visualization of DBLP Network in Gephi

In this task we will use Gephi to visualize parts of the DBLP network, and see some first examples of the analysis capability of Gephi. DBLP is a non-anonymous (with real names) co-authorship network in computer science, publicly available at `dblp.org`, consisting of some three million publications and more than a million authors.

We will be using the Python script `merged_ego_networks.py` to extract "ego-networks" from the DBLP database.

- Work in the subdirectory `TSKS33-sandbox/session1/task1`.
- Select two or three names as seeds, and enter them into the appropriate lines in the DBLP analysis program, `merged_ego_networks.py`. (Hint: add them to the list called `names`.) Any names may be used as seeds. Some suggestions of "famous" scientists in the field can be found as comments in the script.
- Run the `merged_ego_networks`, by typing in the terminal:

  ```
  python3 merged_ego_networks.py
  ```

- Open the output file `DBLP.NET` with Gephi. Make sure to import the network as undirected.
  - How many nodes and links does DBLP have?
  - *Modularity* is a metric used for classification into communities. We will learn about this concept in more detail later in class. For now, just take it for given.

    Assign node color according to modularity class. Also set the node sizes according to degree centrality. Experiment with some different layout algorithms to obtain a figure in the same style as the figure on the cover page of the course material.

    Does the network have a clear community structure?

Task 2: Introduction to the SNAP Python Library

- Work in the subdirectory `TSKS33-sandbox/session1/task2`.
- Run the Python script `generate_examples.py`.
- Go through the script `generate_examples.py`, and explain what each lines does. Check the SNAP-PY 6.0 documentation for the SNAP-specific functions.
- Open the `_Poisson-2.NET` and `_Pref-attach-3.NET` files generated by the script, and create nice visualizations of them with Gephi.
- Look at the degree distributions and information files (`*info.txt`) saved by SNAP.
- For the DBLP, inspect the length of shortest paths between some pairs of nodes. Conclusion?

Task 3: Motifs

- Work in the subdirectory `TSKS33-sandbox/session1/task3`.
- Go through the script `motifs_amazon.py`, and explain what each lines does.
- Use the generated adjacency matrix and the formulas derived in class to compute the number of triangles and wedges. (Tip: You can refer back to this code for future labs!)
- Compare the result given by SNAP with your implementation. Also compare the runtime.

Examination

- Individual oral examination takes place in class (computer lab). Students are also expected to be able to answer questions relating to the course material. All students must study the pertinent course material before coming to the lab.
- Before asking for oral examination, make sure to have all your plots, figures and code readily available. It is suggested to save all plots and code prepared in the lab by collating them into a document, for example, in PowerPoint or LibreOffice.
- Collaboration on this homework in small groups is encouraged, but each student should individually demonstrate understanding of all tasks.