UCLouvain - EPL - LDATA2010 - Project

Implementing a network visualization software

Networks arise in increasingly many situations in data science. As introduced during the exercise session on network visualization, displaying large graphs requires care and is far from being an obvious task. Various algorithms have been designed in the sake of determining the network layout which best reveals its structure.

In this project, you will have the opportunity to create a network visualization software. This includes implementing layout algorithms, enabling the user to change the color and size of the nodes and edges according to some network attributes or properties, allowing the user to interactively navigate in the network, depicting multiple views of the graph structure, etc. Obviously, the aim of the project is not to develop such a complete software as Gephi for instance. Instead, you are asked to implement a few basic features (see Section 1) and to also focus on at least one of the "additional aspects", detailed in Section 2

1 Basic features

You will develop a software with the following features:

1. A user must be able to upload a graph in your software to visualize it and interact with it. You are free to decide the file format that will be used to encode the networks. You may employ several formats.
2. Some example networks will be embedded with your software, to enable a user to load them and get familiar with the features you have implemented. Some examples of possible networks are detailed in Section 4.
3. You will implement at least two layout algorithms to display a network. The user will be able to change the parameters of the algorithms to observe how they change the visualization. For instance, you may implement one or several force-directed layout algorithms, an adjacency matrix view with a heuristic to order the nodes, a circular layout, etc. Some indications are provided in the document `mcguffin-2012-simpleNetVis.pdf`, introduced in Section 4.
4. A user will be able to display multiple views of a network using different layouts, to explore different facets of its structure.
5. You will enable the user to compute some basic properties and metrics of a network, such as the betweenness centrality, the clustering coefficient, minimum spanning trees, the shortest path between a source and target node (both specified by the user), communities among the nodes, etc. To compute these properties, feel free to employ relevant toolboxes, such as `Networkx` in Python. The user will have the possibility to highlight these properties on the graph display, for instance by coloring the shortest path between a source and a target node, coloring the communities, etc.
6. A user will be able to change the color and size of the nodes and edges according to metrics of the network (e.g. node degree, betweenness centrality, etc.), node labels or attributes and edge weights. The drawing of the edges may also modified, by using curved or straight lines.
7. You will enable the user to filter some nodes and edges of the network according to some of their metrics (e.g. degree) or attributes (e.g. edge weight).

You may decide to focus on either directed or undirected graphs, or to manage both.

2 Additional aspects

In addition to the features detailed in Section 1, you will also focus on at least one of the following "additional aspects" while designing your software:

Computational and algorithmic aspect: For this aspect, we expect you to implement several layout algorithms, in particular force-directed ones, to compare them and suggest the ones to use for different tasks. For instance, some algorithms may better scale with large graphs while some other may be better suited for networks with specific structures (e.g. bipartite graphs). Some layout algorithms are referenced in

Gephi documentation. You could also dedicate some layout algorithms for some specific types of graphs (e.g. trees). The focus in this aspect will hence be oriented toward the computational and algorithmic properties of the layout algorithms, and on the identification of the types of graphs for which they are best suited. The final software may thus be employed using a command line.

User interactive aspect: For this aspect, we expect you to implement a user interface enabling navigation through the network for exploratory analysis. For instance, the Fisheye distortion enables focusing on parts of the network when moving the mouse on the screen. The document `lecture_aalto.pdf`, introduced in Section 4, quickly overviews this distortion method and provides a link toward a demo. Other idea's are enabling the user to zoom in the network, to drag some nodes to change their positions using the mouse, to highlight a node selected in one view on all the multiple views of a network, to depict some node label or edge weight information when selecting the corresponding node or edge with the mouse, and any other original type of user interaction.

You are free to choose the additional aspect that you prefer and you can for sure decide to focus on both of them. The aim is that you give some orientation to your software: should it be used as a computational resource or rather as a visual exploration tool, or both? The features that you will implement in the context of the additional aspect(s) are not restricted to the ones mentioned above: any other, possibly original, feature that you could imagine and which is in the spirit of the above aspect(s) may be interesting. If you think of another aspect which is not mentioned above and on which you would like to focus, do not hesitate to ask the teaching assistant during the exercise sessions whether it would be suited to this project.

3 Report

In addition to your software, you are asked to provide a small report (maximum **8 pages**) detailing the features that you have implemented and the additional aspect(s) on which you have been focusing. In particular,

- You can write your report as a user guide for your software.
- Explain and justify your design choices. For which types of graphs do you recommend your software (e.g. trees, general graphs, etc.)?
- Cite the toolboxes, sources and papers that you employed. There is no restriction on the sources you use nor on the paper that you read, but you have to cite them.
- Reasonably detail the layout algorithms that you have employed (e.g. by providing an overview of each one of them without the practical implementation details) and justify why you chose them.
- Provide examples on how to use your software, illustrating its capabilities in terms of scaling, interaction, visualization, etc. (e.g. show a figure with the highlighted shortest between two nodes). You can employ some network examples for this purpose, see Section 4.
- Give some idea's on how to improve your software. Which features might it be worth implementing in future versions? How could you make your software more scalable?

4 Material

To help you getting started, you are provided with the following documents:

lecture_aalto.pdf : lecture slides from the course CS-E4840 at Aalto University. They quickly introduces graph visualization and navigation and give a link toward a demo.

mcguffin-2012-simpleNetVis.pdf : survey paper[1] introducing simple algorithms for network visualization. Some basic methods are detailed and can be easily implemented. References toward further readings are provided.

It is recommended that you start the project by reading these two references.

As to the `data` folder, it contains a text file, `trump-net.txt`, with a network of individuals related to Donald Trump. You can this network as an example to test your software. Other possibilities are the networks in Gephi format employed during the exercise session on network visualization. Many other networks can be found on the web, depending on what you are looking for: large networks to test the scalability of your algorithms, networks with particular attributes for the nodes and edges, networks with particular structures (e.g. bipartite networks, trees) or related to some domain, etc.

[1] Michael J. McGuffin (2012). Simple Algorithms for Network Visualization: A Tutorial. Tsinghua Science and Technology (Special Issue on Visualization and Computer Graphics), Vol. 17, No. 4, 2012, pages 383-398.

5 Practical information

- You can complete the project alone or by groups of 2 students. In case the total number of students is odd, a single group of 3 students is accepted. Obviously, the expected amount of work (in terms of number of implemented features and their complexity) scales with the number of students per group.
- **Programing language**: you can use the one you prefer (Matlab, R, python, etc.), at your best convenience. You can use all the toolboxes, packages, modules, etc., that you find relevant. The only exception is that you must implement the layout algorithms by yourself. Moreover, if you plan to develop an interactive user interface, you can use toolboxes to help you designing it but you can not rely on a already existing one.
- **Deadline for the project submission on Moodle**: Thursday January 3, 15pm. Submit one .zip file per group, containing your report and your codes.
- **Project presentation and discussion**: right after the exam. Each group will have a meeting with the teaching assistant during which the students will first present their software, detail the features that they have implemented and the additional aspect(s) on which they chose to focus. Realizing a small demo of the software would be highly appreciated! After this presentation, a short discussion will take place with the teaching assistant, during which he will ask a few questions on the software and during which the students will explain how they could further improve it.
- Do not hesitate to ask questions to the teaching assistant during the exercise sessions, for instance to define what you plan to implement, the network metrics that you could evaluate, etc.