COSC 2671 Social Media and Network Analytics

Lab Week 10

Community Detection

Learning outcomes:

- Revise community detection concepts
- Compute community detection and other SNA measures of a graph in networkx

Requirements:

- A PC with Internet connection and Python installed (preference is version 3.X, but 2.7 is also okay, but unfortunately we don't have resources to provide support if there are version incompatibility issues)

Resources:

- Associated code in Lab10Code.zip (available on Canvas)

Python Packages Required:

- Network
- python-louvain

Software

- Gephi

Overview

We reuse the Reedit graph we constructed last week and study how to detect communities and evaluate them.

Lab Introduction

In this lab, we use the Reedit graph we built last lab to study how to find communities, visualise them and to evaluate them.

Reddit Remainder

If you haven't the last lab, now it is a good time to complete that. At least register an account and app, and construct a graph from it. In case you haven't done so and wish to do it at a later date, we have included the reply graph **modlab9.graphml** needed for this lab.

Finding Communities

There are many tools available to find communities. We will continue to use networkx, but also explore how it can be done within Gephi.

First, networkx has some built-in community detection algorithms, including CPM. But it doesn't have a modularity maximisation implementation. We will install a package that has an implementation of the Louvain algorithm (to maximise modularity). First, we install the Louvain python package:

```
$ pip install python-louvain
```

Next, open up **redditCommunity.ipynb**. Within it there is skeleton code for loading the graph, including converting the directed graph into an undirected one (both implementations only work for undirected graphs). Next we implement the community detection algorithms.

Exercise:

Go to the CPM documentation (https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.kclique.k_clique_communities.html#networkx.algorithms.community.kclique.k_clique_communities)

It looks very similar to the other networkx functions you have been using. Add code within **redditCommunity.ipynb** to find the communities, there is a **todo** part in the code telling you were to add this. Print out the output of the function (the communities). What does it look like?

Now we call the Louvain algorithm, have a look at the documentation (https://python-louvain.readthedocs.io/en/latest/api.html). Call the relevant function (the relevant package has been included in the header of **redditCommunity.ipynb**). Print out the output of this function (e.g., print(…)). What does it look like?

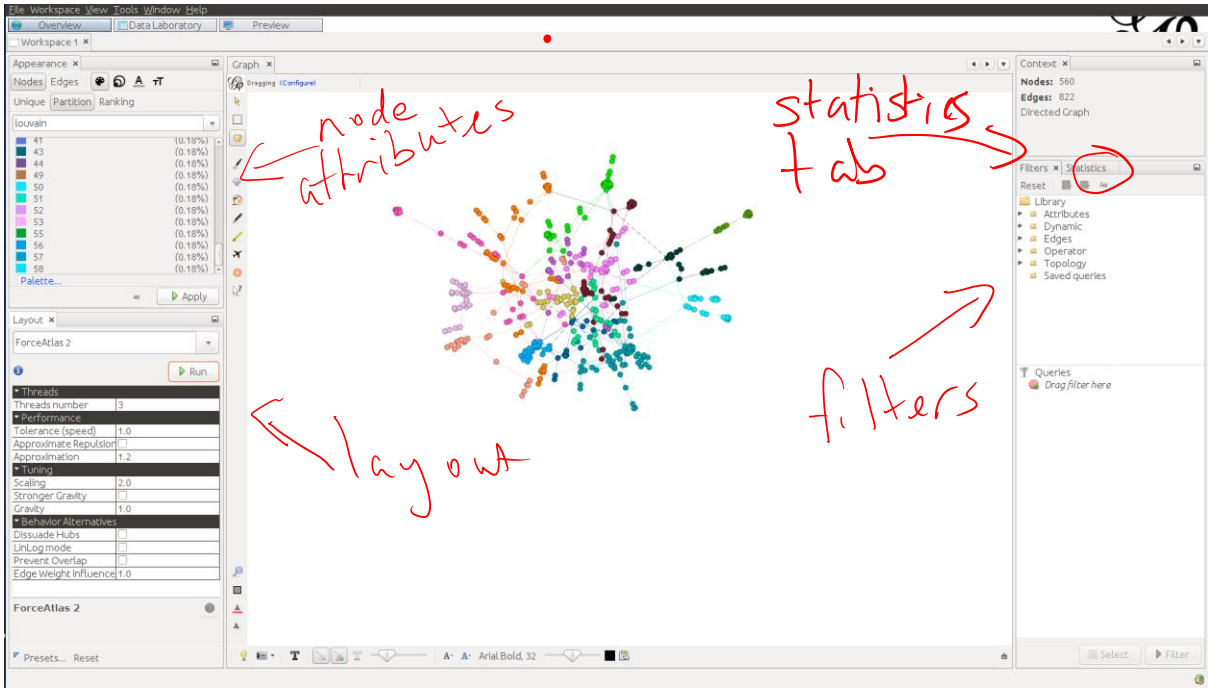
Which set of communities are more appropriate? Do you think you can answer that just from the output of print()?

We also want to visualise the community labels. Similar to the previous lab, we store the additional attributes into the nodes of the reply graph and output as a modified graph. Recall what we did for the Katz and Eigenvector centrality in the previous lab. Do this for the CPM and Louvain labels.

Next we visualise, then try community evaluation methods.

Visualising Communities

Recall that we have been using Gephi to visualise the networks in the last few weeks. We will use Gephi again. Load up the modified graph (with the community labels). Use the node attributes to colour them (see below):



Use the ForceAtlas2 layout, usually better for community structure.

First do it for CPM, then Louvain. Which one appears more "coherent" to your eyes?

Gephi also allows you to filter out some nodes, see the tab in picture below, and play with various options to help improve the visualisation

In addition, Gephi has some analytic tools. Check the statistics tab, under which there is a button to compute modularity and find communities. This actually uses the same approach as networkx to find the communities based on modularity. Try it out. Are there any differences with the ones computed in networkx?

Evaluating Communities

Final part of the lab is to evaluate the communities using one of the measures we learnt in class. Given there is no ground truth labels, we can either construct some attributes and measure or observe their coherence, use visualisation or have a ground truth to compare against. We will do the latter to demonstrate the **purity measure** (the fraction of instances that have labels equal to the community's majority label) we learnt in class, but this is just one way to quantitatively measure the identified communities.

Exercise:

In this lab, we provide a ground truth community. We have already implemented the code to load this ground truth:

```
lGroundTruth = []
# read in ground truth
with open('lab07GroundTruth.csv', 'r') as fIn:
    reader = csv.reader(fIn, delimiter=',')

    for lRow in reader:
        lGroundTruth.append(set(lRow))
```

This loads the ground truth as a list of communities (sets of nodes), e.g., [set('a', 'b'), set('c', 'd')] where 'a' and 'b' are one community, and 'c' and 'd' are another.

This is the same format that both the output of CPM and that we have converted the Louvain output to.

Part of learning Python (and R) is the ability to code up new functionality as needed. networkx nor scikit learn provide a purity measures, so we will code one up. Given the identified community, we want to check the amount of overlap with each ground truth community (each ground truth community can be considered as having the same label). The one with the biggest overlap is the majority class/label, and we sum up the overlap with majority label for each identified community. We then normalise this sum by the number of nodes in the identified communities.

In the code, we have constructed a skeleton of such a function to compute purity (called purity()). Complete its implementation, then evaluate the two sets of identified communities against our ground truth communities. How did the two algorithms perform?