Task 12: Clustering

Step 0: Data

In this practical we will experiment with `facebook_circle.edges`, which is a network of friends for a single Facebook user. The user themselves has been removed, and as a result the graph is not connected.

Step 1: Calculating the number of edges in the graph

Write code to determine the number of edges in the graph. You should get an answer of 2519.

Step 2: Find all connected components

Implement a depth-first search algorithm to check for connectivity and then use it to find all **connected components** in the graph. Two nodes u and v are in the same connected component if and only if there is a path between them. You should find there are 5 connected components in `facebook_circle.edges`.

Step 3: Calculate edge betweenness

Implement code to compute the edge betweenness of all edges in the graph. This is a minor variant of your code from Task 11. The pseudocode is in Section 3.5 of Brandes (2008). However, the psuedocode in the paper predicts betweenness over the union of edges and vertices (i.e. both edge betweenness and node betweenness), but we are interested only in edges, so you will need to adjust it accordingly.

Step 4: Girvan-Newman

Implement the Girvan-Newman algorithm, which divides a graph into clusters by splitting it into relatively dense components. We are following this description of the algorithm, and this is the original paper. In the version we will use, the number of clusters to be returned is specified as a parameter. It works as follows:

1. While the number of connected components is less than the specified number of clusters, and the number of edges in the graph is greater than zero:
    i. Calculate the edge betweenness for every edge in the graph
    ii. Remove edge(s) with the highest betweenness

iii. Recalculate the connected components

In the case where there are ties for highest betweenness, you should remove all the highest betweenness edges. When comparing betweenness values, use a tolerance of 1×10^{-06} for a fuzzy comparison of doubles. This means that the number of connected components you end up with may be higher than the originally specified number.

Starred Tick

In the implementation of Girvan-Newman for the tick, the number of clusters is specified in advance. However, this is a problem if we have no idea of the correct number of clusters. To take this into account, you can implement a 'goodness of clustering' measure called modularity which is described in Newman (2004, page 6). This is equivalent to calculating the complete dendrogram and then choosing the level with the best modularity (see the slides).

Each time we create a cluster in the network, the modularity of the clustering is calculated, and we store the components found at that stage. We run the Girvan-Newman code to completion (i.e. to the point where there are no edges and every node is separate), and then take the set of components associated with the highest modularity.