**Assignment 2**

Graphs Algorithms and Mining

GAM Group 12

|  |  |
| --- | --- |
| **Name** | **BITS ID** |
| D M Pravin Kumar | 2022DA04333 |
| Parveen Kumar |  |

# Question 1

## **Dataset used**

Link : <https://snap.stanford.edu/data/email-Eu-core.html>

**Files:**

* email-Eu-core.txt
* email-Eu-core-department-labels.txt

## Approach

1. Create a Directed Graph (nx.DiGraph) from the file **email-Eu-core.txt**. The file contains the edge list in the format (send\_node received\_node)

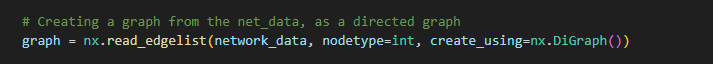


Figure 1: Creating the graph using Directed edges

1. Identify the largest weakly connected component.
   1. The largest weakly connected component can be directly identified by using the maximum of nx.weakly\_connected\_components with the key set to len (function)

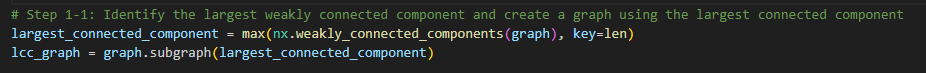


Figure 2: Getting the largest weakly connected component

1. Compute the communities in the largest weakly connected component using Louvain Community Detection Algorithm.
   1. Using the Louvain community detection algorithm to get the communities



Figure 3: Computing the communities

1. Get Ground Truth Communities
   1. Reading the file email-Eu-core-department-labels.txt will give the node and the department label as a set of list separated by spaces(node department)
   2. Create a dictionary of the format {dept:[nodes]} which will contain all the nodes that are against a label provided in the dataset.
   3. Make a sorted list of all the dictionary values. The output of this is a list of lists, each list is a community and all the communities are sorted in order.

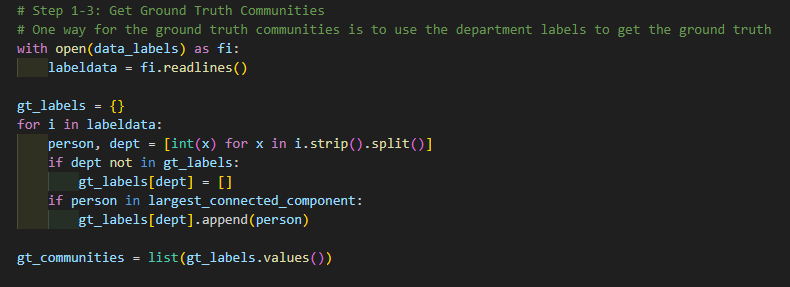


Figure 4: Getting the ground truth communities

1. Compare communities with Ground Truth Communities
   1. For all the nodes that are present in the set of the largest weakly connected components, check the index of the community in which the node is present and for each of the nodes in the largest weakly connected component make two lists, one for the computed communities and one for the ground truth communities.

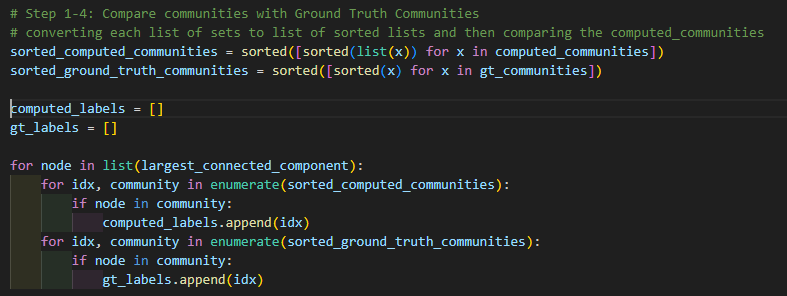


Figure 5: Making labels for both GT and generated communities

* 1. Now using the metrics provided by the sklearn.metrics.cluster module finding the following scores, rounded to 5decimal places:
     1. **Fowlkes mallows score [2]**
     2. **Adjusted mutual information score [3]**

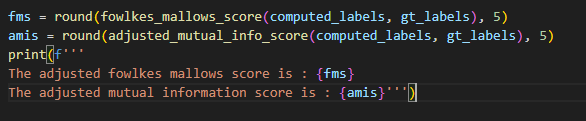


Figure 6: Code to compute the scores

The output is as below:



Figure 7: Output of computed scores

## Justification

The scores of Adjusted mutual information [3](value of 0.56419) and Fowlkes mallows score [2]

(geometric mean of Precision and Recall) is 0.438.

While the ideal values for both the scores are 1 and the worst case values are 0. The values of 0.438 and 0.56419 respectively give us an indication that there are some of the nodes that are clustered properly into correct communities but there are many other nodes that are not properly clustered. One of the reasons could be the number of communities formed by the Louvain clustering method (8) is way different from the number of clusters from the ground truth.

The final conclusion that there is a notable correlation between the identified communities and the ground truth communities and that there is a scope of improvement possible for more robust community detection methodology.

# Question 2

One of the community detection algorithms that are used is Louvain community detection algorithm [1]. This algorithm works in two phases:

## Description of the Algorithm

**Start:**

All the nodes are in their own communities.

**First Phase:**

For each node:

* Find the neighbours, and compute the modularity gain if the node is placed in the community of adjacent node.
* Find the community with maximum modularity gain
* If maximum gain > 0 then place the node in the community of the neighbour
* Repeat the steps until no node changes the community

**Second Phase:**

* A new network is created with the communities created in first phase.

The phases are iteratively repeated until there is no more modularity gain (or gain is less than threshold)

This two phased approach is important as the number of communities drop very drastically and the computation is generally done in **O(nlogn)** complexity

## Approach of the program:

Input : Adjacency Matrix

We define two functions:

First function to compute the modularity gain from adjacency matrix:

* Create a zeros matrix of the same shape as the adjacency\_matrix.
* Define all the variable and compute using the formula below

Mod gain = (k\_in)/(2\*m) – ((sigma\_tot \* k\_i) / (2\*m2))

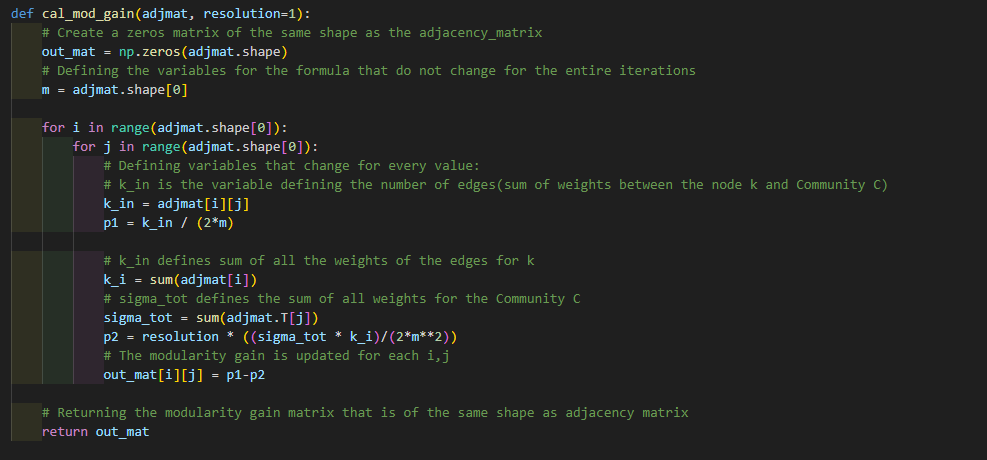


Figure 8: Code for calculating modularity gain

Second function is defined to generate the Louvain Communities

The two phase approach as described above is coded in the program, comments are provided appropriately explaining every step of the code.

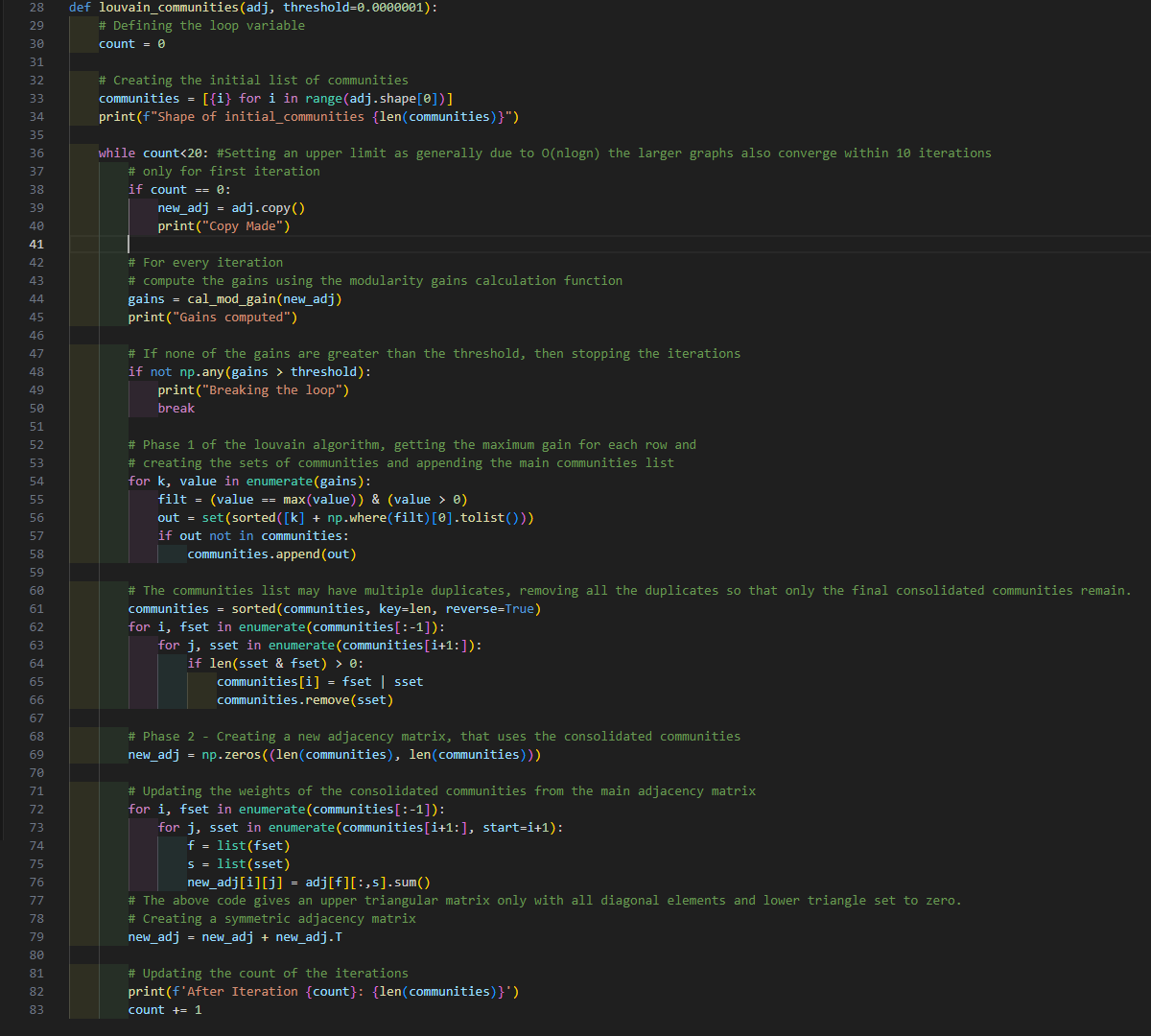


Figure 9: Code for computing louvain communities (For undirected graphs only)

## Output

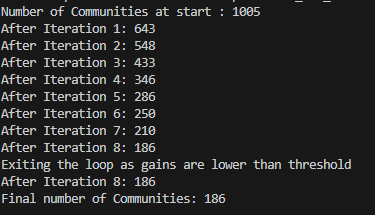


Figure 10: Communities from the data

# References

[1] “DiGraph—Directed graphs with self loops — NetworkX 2.8.7 documentation,” *networkx.org*. <https://networkx.org/documentation/stable/reference/classes/digraph.html>

[2] “sklearn.metrics.fowlkes\_mallows\_score — scikit-learn 0.23.1 documentation,” *scikit-learn.org*. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fowlkes_mallows_score.html>

[3] “sklearn.metrics.adjusted\_mutual\_info\_score,” *scikit-learn*. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_mutual_info_score.html#sklearn.metrics.adjusted_mutual_info_score>

[4] “SNAP: Network datasets: email-Eu-core network,” *snap.stanford.edu*. <https://snap.stanford.edu/data/email-Eu-core.html>

[5] “louvain\_communities — NetworkX 3.2.1 documentation,” *networkx.org*. <https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.louvain.louvain_communities.html>