# Comparison of Community Detection Algorithms in Python using Ground-Truth Communities

Swetha P

December 2024

## 1 Introduction

A graph consists of vertices (or nodes), which represent entities, and edges (or links), which represent connections or relationships between these entities. Graphs can be directed or undirected. A network is a real-world system modeled as a graph. Some examples of real-world systems that are modeled as networks are social networks, biological networks, transportation networks, communication networks, etc. The nodes in a network generally contain additional information, such as labels, weights, or attributes.

An important step in studying a network is identifying the communities present within it. Communities are defined as sets of nodes in a network that are densely connected to each other and distinct from other nodes in the network. In real-world networks, a community represents a group of interacting entities. For example, on a social media network, students from a particular college may form a community if they interact frequently with each other. Identifying the communities in networks has several applications, such as:

- Identifying social groups
- Customer segmentation
- Recommendation systems for suggesting friends, people to follow, etc.
- Gene regulation networks to identify genes that work together to express some traits
- Epidemiology
- Citation networks to identify collaboration patterns
- Power grids

In fields like epidemiology, community detection plays a very important role in preventing the spread of diseases by identifying the possible contacts of an infected person. Therefore, it is important to develop methods for community detection. Several algorithms have been developed to detect communities in networks such as the Louvain algorithm, Leiden algorithm, Girvan-Newman algorithm, etc. These algorithms have strengths and weaknesses. Generally, communities are non-overlapping, i.e., a node belongs to only one community.

The communities that are discovered by these community detection methods reveal underlying patterns in the data. However, it must be noted that the community detection methods only use the intrinsic properties of the network such as degree of a node, edge connections, etc. Generally, once the communities are found, the next step is to identify an external property that explains the community structure. For example, in the communities in a music streaming platform can correspond to people who like similar genres of music. In real-world systems it can be difficult to identify a single factor that can explain the communities that are detected.

Since, there is no fixed definition of how a community should be defined, there is some ambiguity in choosing one particular method for partitioning the networks and also for evaluating the results of the partitioning. Generally, we choose a particular metric and apply an algorithm to find the community partitions that optimizes this metric. One of the metrics used to evaluate the results of the community detection algorithms is called modularity. It is also necessary to compare the detected communities with real-world clusters which are called "ground-truth communities". Metrics are also needed to compare the communities detected by different community detection algorithms.

In this report, we describe modularity and other metrics used to evaluate and compare the results of the community detection. We also describe some commonly used community detection methods that are implemented in Python. Finally, we apply the different algorithms to some datasets which already have some ground-truth communities' information and evaluate the results.

# 2 Objective of the study

The objective of our study is to perform community detection on various datasets with known ground-truth communities using different algorithms and evaluate the performance of these algorithms using the metrics modularity, normalized mutual information, variation of information, and adjusted rand index.

# 3 Evaluation Metrics

## 3.1 Modularity

Modularity (Q) [1] is a measure of the strength of the partition of the network into communities. Modularity is defined as difference between the number of edges within a community and the expected number of such edges in a random graph with the same degree of nodes.

$$Q = \frac{1}{2m} \sum_{i,j} (A_{i,j} - \frac{k_i k_j}{2m}) \delta(c_i, c_j), \tag{1}$$

where,
$A_{i,j}$ - wight of the edge between nodes $i$ and $j$
$k_i$ - degree of nod $i$
$m$ - total umber of edges in the graph
$c_i$ - community of node $i$
$\delta(c_i, c_j)$ - 1 if $c_i = c_j$, 0 otherwise
Modularity values lies in the range $[-1, 1]$. A positive value close to 1 indicates a good community structure is detected. A positive value close to zero indicates that the detected communities are of poor quality. Negative values mean that the detected communities are worse than random.

Modularity optimization is used as a tool to detect communities in some algorithms such as Louvain and Leiden algorithms. Modularity can also be used to evaluate the quality of partitions. A major limitation of using modularity optimization to detect communities is the resolution limit. These algorithms struggle to find smaller communities and end up merging them into larger groups.

### 3.2 Normalized Mutual Information (NMI)

Normalized Mutual Information (NMI) [2] is used to compare the similarity between two different partitions of the same graph. Given two partitions A and B, NMI is calculated using the formula:

$$NMI = \frac{2I(A.B)}{H(A) + H(B)}, \tag{2}$$

where, $I(A, B)$ is the shared information between two partitions
$H(A)$ is the entropy of partition A
$H(B)$ is the entropy of partition B
NMI has values in the range [0,1]. An NMI value of 1 indicates identical partitions. An NMI value of 0 indicates that the two partitions are completely independent and do not have any of the same information. Therefore, a value close to 1 indicates highly similar partitions and a value close to 0 indicates highly dissimilar partitions. NMI is not affected by the size of the graph or the number of partitions. Similarly, it is not affected by the order of the partitions.

### 3.3 Variation of Information (VI)

Variation of Information (VI) [3] is a similarity measure. Given two partition A and B,

$$VI = H(A) + H(B) - 2I(A, B), \tag{3}$$

where, $I(A, B)$ is the shared information between two partitions
$H(A)$ is the entropy of partition A
$H(B)$ is the entropy of partition B
VI is a non negative value. A small value of VI indicates that the two partitions share a lot if information and hence, are similar. Thus, we can compare the similarity or dissimilarity between different sets of partition. However, it is difficult to interpret the value of VI in absolute terms.

## 4 Adjusted Rand Index (ARI)

The Adjusted Rand Index (ARI) [4] is a statistical measure that is used to measure the similarity between two partitions. Given two partitions A and B,

$$ARI = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}, \tag{4}$$

where,
Index is the pairs of nodes that are in the same cluster in both A and B and the pairs of nodes that are in different clusters in bot A and B
Expected Index is the expected agreement in a random graph with the same degree for each node
Max Index is the total number of pairs of nodes in the dataset
ARI values lie between -1 and +1, with +1 indicating a perfect agreement between the partitions and 0 indicating that the partitions are dissimilar.

# 5 Algorithms for Community Detection

## 5.1 Louvain Algorithm

The Louvain Community Detection algorithm is an algorithm that detects non-overlapping communities from graphs. The Louvain algorithm was created by Blondel et al [5] This is a modularity optimization algorithm. It works in two phases. In the first phase, the nodes are repeatedly moved to neighbouring communities and modularity is checked. The process is repeated till the modularity value no longer increases. In the second phases, each community is considered as a single supernode, the edge between the supernode is the sum of all the edge weights between the communities. The two phases are repeated till no further increase in modularity is possible. A major disadvantage of the Louvain algorithm is the formation of badly connected communities due to various reasons. In python, the Louvain algorithm can be implemented using the python-louvain library. The library performs community detection and also has function to evaluate the modularity of a given partition.

## 5.2 Leiden Algorithm

The Leiden algorithm is another modularity optimization based community detection algorithm. It was developed by Traag et al [6] to overcome the disadvantages of the Louvain algorithm namely, badly connected communities and poor resolution limit. The Leiden algorithm includes an additional refinement phase between the two primary phases of the Louvain algorithm described in 5.1. In python, the Leiden algorithm can be implemented using the leidenalg library.

## 5.3 Newman's Leading Eigenvector Algorithm

A major drawback of the Louvain and Leiden community detection algorithms is that we cannot specify the number of communities that the network should contain. This prevents us from accurately comparing the results from these algorithms with the ground-truth data, as the number of detected communities might be different from the number of ground-truth communities. Therefore, we consider Newman's Leading Eigenvector algorithm [1] which allows us to specify the number of expected clusters prior to implementation. This algorithm involves building a modularity matrix, then splitting the network into along the leading eigenvector of the modularity matrix. The process is iterated until the expected number of partitions are achieved. The disadvantage of this algorithm is its computational complexity. So, it can only be applied to small and medium-sized networks. In python, the igraph library has a function to implement this algorithm.

## 5.4 K-means clustering

We have also used a manual Graph Neural Network (GNN) implementation to perform community detection using the unsupervised learning technique called k-means clustering. In this method, we can fix the number of communities at the beginning. We have used the pytorch-geometric library to train a Graph Neural Network (GNN) for node representation learning, followed by clustering the learned embeddings using k-means clustering. The GNN, defined with two graph convolutional layers (GCNConv), computes node embedding using node features and the edge structure of a graph. we use cross-entropy loss function to train the GNN. The objective is clustering node embeddings to maximize structural similarity. After training, the embeddings are clustered into teh specified number of communities. Modularity is calculated to assess the quality of these communities.

# 6 Results

All datasets used in the study have been obtained from [7]. We have performed community detection on 3 datasets using the algorithms described in 5. Then, we find the modularity of the partition for each algorithm. We also find the NMI, VI, and ARI scores by comparing the results of each algorithm with the ground-truth community partitions mentioned in the datasets. In this way, we can compare which algorithm produces communities which closely resemble the ground-truth communities.

## 6.1 CoRA Dataset

The CoRA dataset is a citation network dataset consisting of 2708 nodes and 5278 edges. The nodes represent scientific publications and the edges represent how they cite each other. The nodes belong to one of 7 classes with labels such as "Genetic Algorithms", "Reinforcement Learning", etc.

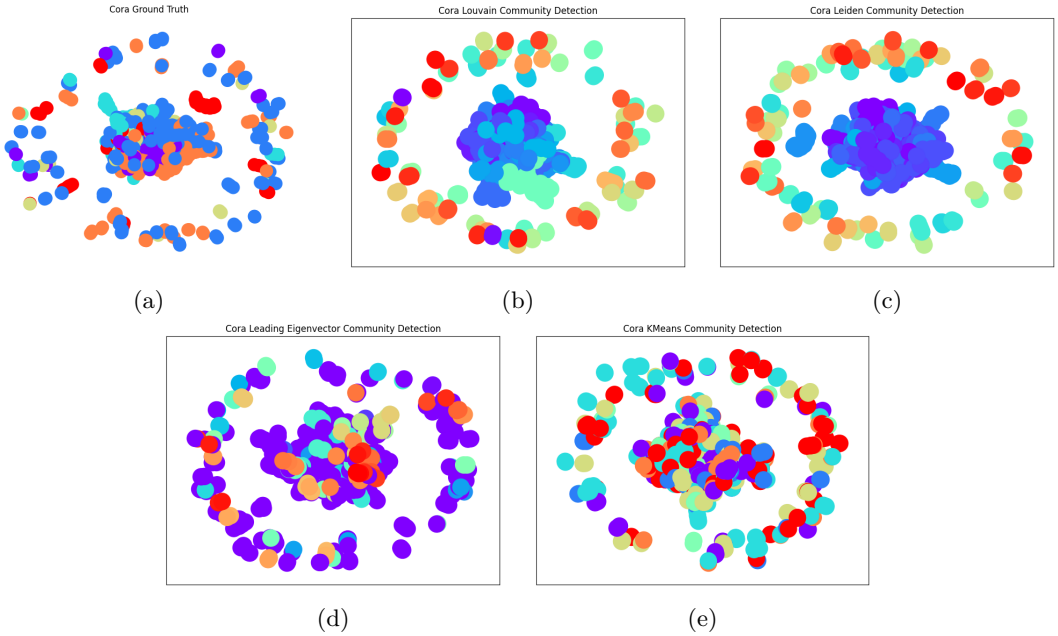


Figure 1: CoRA dataset network with nodes belonging to different communities shown in different colors.

| Algorithm | Modularity | NMI | VI | ARI | No. of Communities |
|---|---|---|---|---|---|
| Louvain | 0.8132 | 0.1609 | 4.3038 | 0.0485 | 103 |
| Leiden | 0.8215 | 0.2615 | 3.7913 | 0.1339 | 105 |
| Leading Eigenvector | 0.0265 | 0.1604 | 2.0598 | 0.0365 | 7 |
| K-means Clustering | 0.3300 | 0.0531 | 3.5749 | 0.0358 | 7 |

Table 1: Comparison of community detection algorithms based on Modularity, NMI, VI, and ARI for the CoRA dataset.

## 6.2 AIML dataset

The AIML dataset is also a citation network dataset consisting of 3445 nodes and 10761 edges. The nodes represent scientific publications in the Artificial Intelligence and Machine Learning domain and the edges represent how they cite each other. The nodes belong to one of 7 classes with labels such as "Genetic Algorithms", "Reinforcement Learning", etc.

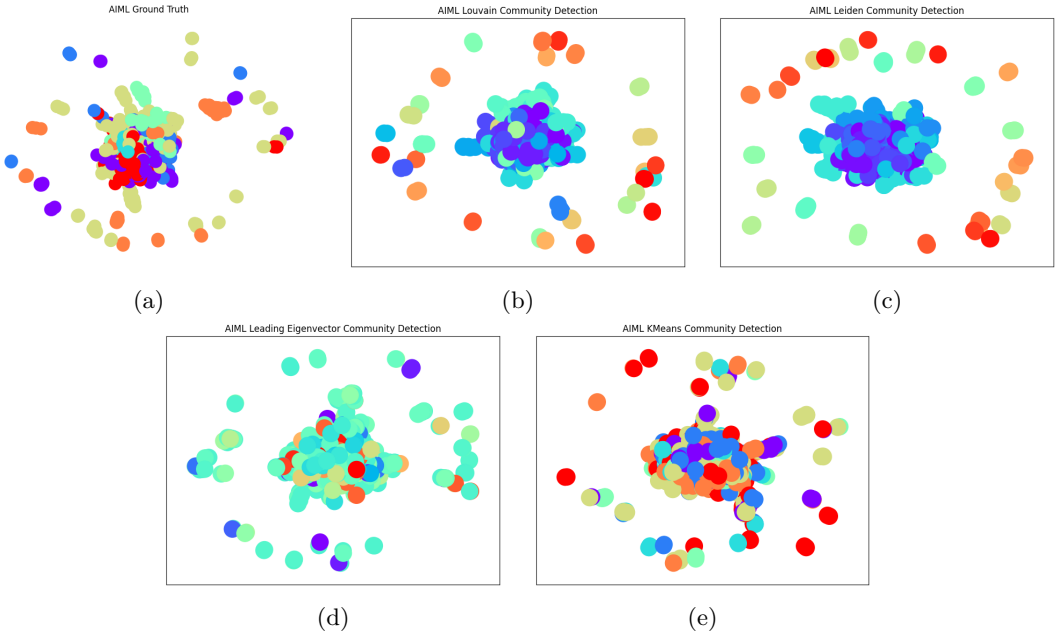


Figure 2: AIML dataset network with nodes belonging to different communities shown in different colors.

| Algorithm | Modularity | NMI | VI | ARI | No. of Communities |
|---|---|---|---|---|---|
| Louvain | 0.7496 | 0.0334 | 4.5619 | 0.0047 | 52 |
| Leiden | 0.7536 | 0.1229 | 4.1512 | 0.0378 | 52 |
| Leading Eigenvector | 0.0319 | 0.1346 | 3.8192 | 0.0442 | 7 |
| K-means Clustering | 0.2796 | 0.0060 | 3.7676 | 0.0016 | 7 |

Table 2: Comparison of community detection algorithms based on Modularity, NMI, VI, and ARI for the AIML dataset.

### 6.3 Football dataset

The football dataset represents a network of American college football games in the year 2000. The nodes represent different colleges and the edges represent games between two colleges in that season. There are 115 nodes and 613 edged. The nodes belong to 12 classes representing the region of USA where the college is present.

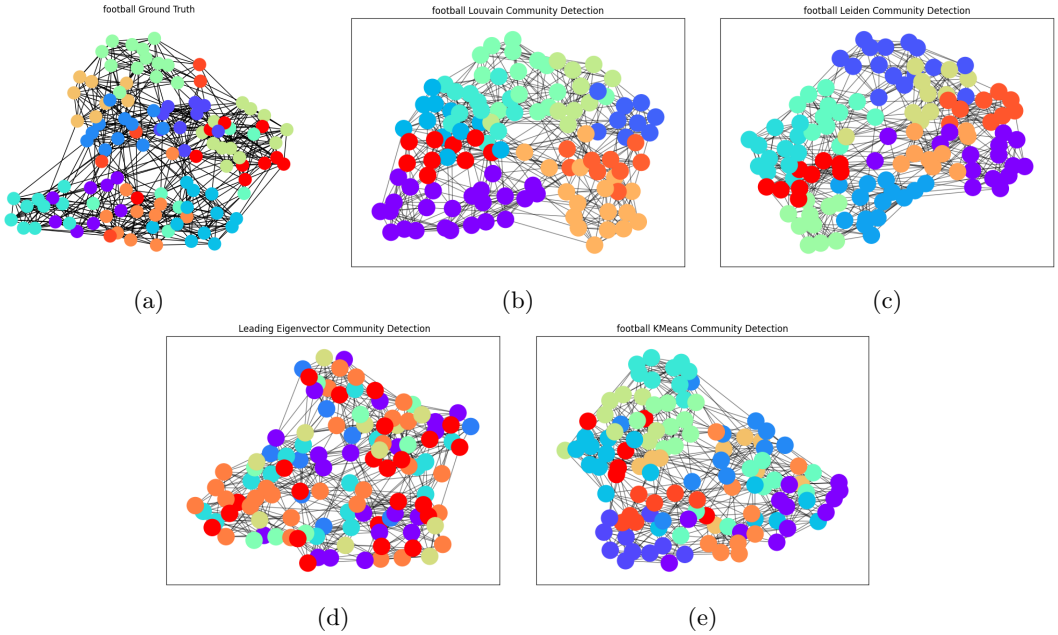


Figure 3: football dataset network with nodes belonging to different communities shown in different colors.

| Algorithm | Modularity | NMI | VI | ARI | No. of Communities |
|---|---|---|---|---|---|
| Louvain | 0.6042 | 0.2021 | 3.6651 | 0.0017 | 9 |
| Leiden | 0.6046 | 0.4679 | 2.5203 | 0.2069 | 10 |
| Leading Eigenvector | 0.0123 | 0.3838 | 2.6562 | 0.1421 | 12 |
| K-means Clustering | 0.3588 | 0.2949 | 3.4627 | 0.0304 | 12 |

Table 3: Comparison of community detection algorithms based on Modularity, NMI, VI, and ARI for the Football dataset.

# 7 Conclusion

From the results of our study we see that the Louvain and Leiden algorithms always give the partitions with the best modularity values. This is expected because these are modularity optimization algorithms. The Leiden algorithm gives marginally better modularity value than Louvain in all three cases. But, in the CoRA and AIML datasets, the number of communities detected by these algorithms is very high compared to the number of ground-truth communities. Newman's leading eigenvector method produces the worst modularity score in all the datasets. However, Newman's leading eigenvector method produces the least VI scores. The Louvain algorithm performs poorly on the ARI index. By examining all the metrics, we conclude that the Leiden algorithm performs consistently well across all the metrics. However, it must be noted that although the Leiden algorithm performs better compared to the other algorithms, the similarity between the detected communities and the ground-truth communities is very low in all the cases. This indicates the need for further understanding the reason for the emergence of these communities. It is possible that the community detection algorithms are picking up hidden structures that are not explained by the simple ground-truth partitions. This is also supported by the fact that the Louvain and Leiden algorithms detect more number of communities as compared to the given data. In future, the study can be expanded to networks of different sizes and different community detection algorithms to compare the results of community detection from networks to ground-truth data.

# References


[1] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006.

[2] Pablo A. Estevez, Michel Tesmer, Claudio A. Perez, and Jacek M. Zurada. Normalized Mutual Information Feature Selection. *IEEE Transactions on Neural Networks*, 20(2):189–201, February 2009.

[3] Jorge M. Santos and Mark Embrechts. On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification. In Cesare Alippi, Marios Polycarpou, Christos Panayiotou, and Georgios Ellinas, editors, *Artificial Neural Networks – ICANN 2009*, pages 175–184, Berlin, Heidelberg, 2009. Springer.

[4] Marina Meilă. Comparing Clusterings by the Variation of Information. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Learning Theory and Kernel Machines*, pages 173–187, Berlin, Heidelberg, 2003. Springer.

[5] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008.

[6] V. A. Traag, L. Waltman, and N. J. van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, March 2019.

[7] Vladimir Ivashkin and Pavel Chebotarev. Do logarithmic proximity measures outperform plain ones in graph clustering? In *International Conference on Network Analysis*, pages 87–105. Springer, 2016.