CDlib: a Python Library to Extract, Compare and Evaluate Communities from Complex Networks

Giulio Rossetti[1], Letizia Milli[2], and Rémy Cazabet[3]

[1] KDD Lab. ISTI-CNR, via G. Moruzzi, 1 Pisa, Italy
[2] University of Pisa,
Largo Bruno Pontecorvo, 2 Pisa, Italy
[3] Université de Lyon, Lyon, France

1 Introduction

In the last decades, the analysis of complex networks has received increasing attention from several, heterogeneous fields of research.

One of the hottest topics in network science is Community Discovery (henceforth **CD**), the task of clustering network entities belonging to topological dense regions of a graph.

Although many methods and algorithms have been proposed to cope with this problem, and related issues such as their evaluation and comparison, few of them are integrated into a common software framework, making hard and time-consuming to use, study and compare them. Only a handful of the most famous methods are available in generic libraries such as NetworkX and Igraph.

To cope with this issue, we introduce a novel library designed to easily select/apply community discovery methods on network datasets, evaluate/compare the obtained clustering and visualize the results.

This extended abstract is a shorter version of the full paper of the same name published in Applied Network Science in December 2019 [1]

2 CDlib: Community Discovery Library

We designed CDlib - *"(C)ommunity (D)iscovery Library"* - to simplify the definition/execution/evaluation of community discovery analysis. CDlib is a Python package built upon the network facilities offered by NetworkX and Igraph. The library, available for Python 3.x, is currently hosted on GitHub[4], on pypi[5] and has its online documentation on ReadTheDocs[6].

At the date of publication, CDlib provides **37** implementations of CD algorithms, including **14** overlapping, **1** fuzzy, and **2** edge partitions methods.

[4] CDlib GitHub: `https://goo.gl/Gu3VSV`
[5] CDlib pypi: `https://goo.gl/FPtHHU`
[6] CDlib docs: `https://goo.gl/ggGbUz`

2.1 Library Rationale

The library provides several community detection algorithms, implemented such as (i) they take as input a unified graph topology representation (networkx or igraph network object) (ii) they return a clustering using a unified representation.

The standardization of clustering representation - and the decoupling of input/output w.r.t. algorithmic implementations - makes easy to extend CDlib with novel algorithms.

Once computed the desired network clustering, CDlib allows its users to: (1)evaluate it using several fitness scores (2) compare it with alternative partitions (3) visualize it using predefined and standard graphic facilities.

Code snippet 1.1 shows an example of computing communities, evaluating their quality, comparing partitions, and visualizing the result using CDlib

```
from cdlib import algorithms, viz
import networkx as nx

# Network topology
g = nx.karate_club_graph()

# Models execution
louvain_coms = algorithms.louvain(g)
leiden_coms = algorithms.leiden(g)

# Modularity evaluation
louvain_mod = louvain_coms.erdos_renyi_modularity()
leiden_mod = leiden_coms.erdos_renyi_modularity()

# Clustering comparisons
nmi = louvain_coms.normalized_mutual_information(leiden_coms)

# Visualization
viz.plot_community_graph(g, louvain_coms)
```

Code example 1.1. Code example.

2.2 Network Clustering

CDlib implements 37 published CD algorithms, using authors' implementations when possible, and some re-implementations. Due to the limited space, the complete list can be found in the original article [1] or in the documentations.

The library explicitly handles three different sub-types of clustering:

- *Partitions* (Crisp clustering): each node (edge) belongs to a unique cluster (community);
- *Overlapping*: each node (edge) is allowed to belong to more than one community at the same time;
- *Fuzzy*: each node (edge) belongs to multiple communities with different level of involvement in each one of them.

Name	Description
average_internal_degree	The average internal degree of the community set.
conductance	Fraction of total edge volume that points outside the algorithms.
cut_ratio	Fraction of existing edges (out of all possible edges) leaving the community.
edges_inside	Number of edges internal to the community.
expansion	Number of edges per community node that point outside the cluster.
fraction_over_median_degree	raction of community nodes having internal degree higher than the median degree value.
[2]	
internal_edge_density	he internal density of the community set.
normalized_cut	Normalized variant of the Cut-Ratio.
max_odf	Maximum fraction of edges of a node of a community that point outside the community itself.
avg_odf	Average fraction of edges of a node of a community that point outside the community itself.
flake_odf	Fraction of nodes of the clustering that have fewer edges pointing inside than to the outside of their communities.
significance	Estimate the likelihood that the identified partition appears in a random graph.
size	Number of community nodes.
surprise	Statistical approach that assumes that edges emerge randomly according to a hyper-geometric distribution: the higher the surprise, the less likely the clustering is resulted from a random realization.
triangle_participation_ratio	Fraction of community nodes that belong to a triad.
newman_girvan_modularity	Difference of the fraction of intra community edges of a clustering with the expected number of such edges if distributed according to a null model.
erdos_renyi_modularity	Variation of the Newman-Girvan modularity that assumes that nodes in a network connected randomly with a constant probability p.
link_modularity	Variation of the Girvan-Newman modularity for directed graphs with overlapping communities.
modularity_density	Variation of the Erdos-Renyi modularity that includes information about community sizes into the expected density coefficient so to avoid the negligence of small and dense communities.
z_modularity	Variant of the standard modularity proposed to avoid the resolution limit.

Table 1. Fitness functions implemented in CDlib.Upper-part: community wise scores. Lower part : modularity-based quality scores.

Name	Description
ami	Adjusted Mutual Information is an adjustment of the Mutual Information score to account for chance.
ari	The Rand Index computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings.
f1	Average F1 score (harmonic mean of Precision and Recall) of the optimal matches among the partitions in input. clustering.
nf1	Normalized version of F1 that corrects the resemblance score taking into account degree of node overlap and clutering coverage.
nmi	Normalized Mutual Information (NMI) is an normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation)
onmi	Extension of the Normalized Mutual Information (NMI) score to cope with overlapping partitions.
omega	Resemblance index defined for overlapping, complete coverage, clusterings.
vi	Variation of Information among two nodes partitions.

Table 2. Clustering comparison functions implemented in CDlib.

2.3 Clustering Evaluation and Comparison

CDlib allows not only to compute network clusterings applying several algorithmic approaches but also enables the analyst to characterize and compare the obtained results.

Clustering evaluation and comparison facilities are delegated to the `cdlib.evaluation` submodule. It provides several fitness scores, listed in Table 1, as well as clustering comparison measures, reported in Table 2.

2.4 Visualization Facilities

To allow the final user visualising clustering results, CDlib exposes a set of predefined visual facilities using Matplotlib. These facilities are exposed through the visualization submodule `cdlib.viz`. Such submodule offers two different classes of visualization:

Network Visualization, that allows plotting a graph with node color coding for communities

Analytics plots, where community evaluation outputs can be easily used to generate a visual representation of the main partition characteristics (Figure 1).

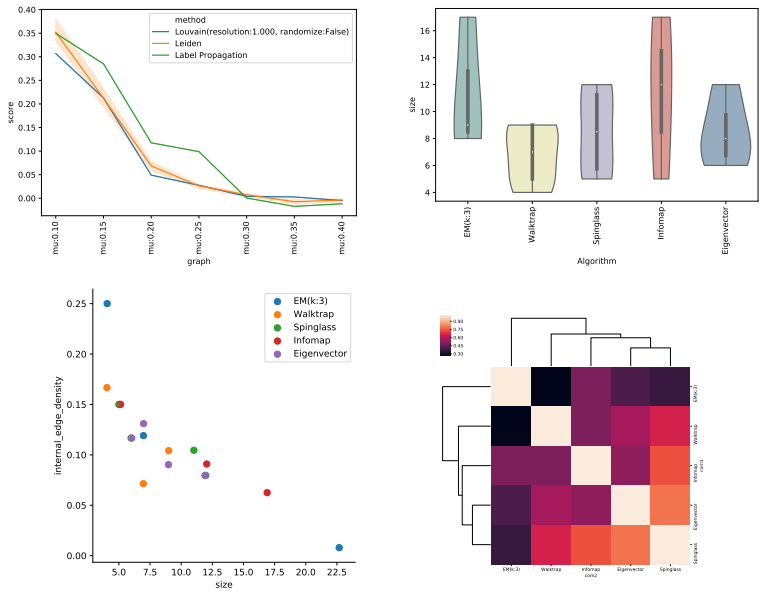

Fig. 1. Visual Analitycs. Comparing (1) several algorithms on several networks, (2)properties of communities, (3)relation between properties, (4)Similarity between clustering.

3 Conclusions and Future Works

CDlib is an ongoing open project: we plan to further extend it by integrating novel algorithms (contributions are welcome), by supporting alternative clustering definitions (i.e., multiplex, evolving,. . .) and by integrating evaluation/visualization facilities.

References

1. Rossetti, G., Milli, L., Cazabet, R.: Cdlib: a python library to extract, compare and evaluate communities from complex networks. Applied Network Science **4**(1), 52 (2019)
2. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems **42**(1), 181–213 (2015)