TP on Community Detection

andrea.failla@phd.unipi.it

November 2024

`cdlib` is a python library designed to provide support for extracting, analyzing and comparing network clusterings. `cdlib` is mostly developed and maintained by Giulio Rossetti (ISTI-CNR, Italy) and Remy Cazabet (Univ. Lyon 1, France). You can install it using pip

```
$pip install cdlib
```

This requires at least python 3.7. The library offers 90+ community discovery algorithms organized into four families:

- Crisp (non-overlapping) communities
- Overlapping communities
- Fuzzy communities
- Node-attributed communities
- Communities on bipartite networks
- Link Communities
- Temporal communities

The library is well documented. Take a look in case you don't know how to proceed. `https://cdlib.readthedocs.io/en/latest/reference/reference.html`.

1 Identifying Communities

1. Download data about co-acting relations during the first season of Game of Thrones

   ```
   $wget https://andreafailla.github.io/uploads/data/got-s1-edges.csv
   ```

2. Read the network with the following function and explore the network's basic properties: number of nodes, edges, density, and clustering coefficient.

```
def read_net_w(filename):
    g = nx.Graph()
    with open(filename) as f:
        f.readline()
        for l in f:
            l = l.split(",")
            g.add_edge(l[0], l[1], weight=int(l[2]))
    return g
```

3. Use the Louvain algorithm to detect communities in the dataset:

```
from cdlib import algorithms
c_louv = algorithms.louvain(g)
c_louv.communities # list of communities
```

4. Find the number of communities and their sizes

5. louvain's default `resolution` is 1.0. Change the parameter and check how the number/size of the communities change

6. visualize the community structure using cdlib's built-in viz utility:

```
from cdlib import viz
pos = nx.spring_layout(g)
viz.plot_network_clusters(g, communityobject, pos,
figsize=(20, 20))
```

7. You can also visualize communities as "meta-nodes", i.e., a graph where each node represents a community:

```
viz.plot_community_graph(g, communityobject,
figsize=(10, 10))
```

1.1 Internal Evaluation

1. Compute communities with at least two other non-overlapping algorithms.

2. in the `evaluation` module, you will find several quality functions to evaluate graph clustering, e.g., modularity, size, internal edge density... Use the `viz.plot_com_properties_relation` method to compare the various methods on size and internal edge density. (Check the documentation)

3. Which community algorithm returns the partition with the highest modularity? Why do you think that's the case?

1.2 External/Qualitative Evaluation

- We will use external metadata about the characters' houses. You can download the data with:

  `$wget https://andreafailla.github.io/uploads/data/got-s1-attrs.csv`

  Then, use the first function to read the house data, and the second to compute purity values for the communities in your partitions. Which algorithm returns the purest communities (on average)?

```
def read_houses(filename):
    node_to_house = {}
    with open(filename) as f:
        f.readline()
        for l in f:
            l = l.rstrip().split(",")
            node_to_house[l[0]] = l[2]
    return node_to_house
```

```
from collections import Counter
def community_purity(coms, attributes):
    purities = []
    for c in coms.communities:
        houses = []
        for node in c:
            if node in attributes:
                houses.append(attributes[node])

        cnt = Counter(houses)
        purity = max(cnt.values())/sum(cnt.values())
        purities.append(purity)
    return purities
```

- Let's pretend that the communities identified by Louvain (resolution 1.0) are the "ground truth". Compute NMI to find the which partition is closer to the Louvain one (except louvain, of course).