PROJECT: COMMUNITY DETECTION

Overview: Community detection is a ubiquitous graph data mining task. Depending on the type of graphs (e.g., directed, undirected, static, dynamic, weighted, unweighted, labeled, unlabeled, unipartite, bipartite or multipartite) and depending on the goal of the graph mining task, community detection can be differently defined. Moreover, for a given community definition, various community detection algorithms could be designed and implemented that target either dense or sparse graphs and meet particular performance characteristics. The performance could be measured along various dimensions: time and/or space efficiency, accuracy of identifying the communities with the known ground truth, etc.

Goal: To implement the given community detection algorithm for real-world graphs.
To do that, your team will need to:

1. Implement the algorithm assigned to your team with an "obsession" with the highest performance possible.
2. Provide a detailed analysis of the performance of your algorithm/implementation.

Input: You will be provided with the following materials in advance:

- Graph datasets with groundtruth communities:
  - Graph 1: Amazon.
  - Graph 2: DBLP.
  - Graph 3: YouTube.
- Scientific publication that describes the algorithm to be implemented.
- Benchmarking codes to measure the performance of your team's community detection algorithm with respect to various performance characteristics.

  NOTE: For each type of graph (Amazon, DBLP and YouTube), there are 4 graphs of different sizes with ground truth communities: a small graph (≈2,500 nodes), a medium graph (≈5,000 nodes), a large graph (<100,000 nodes), and the original graph (>300,000 nodes). Please note that you do not have to run experiments using graphs of all sizes. Selecting one size for each type of graph is enough. Make sure you specify on your report which size you used for each graph.

Output: The implementation of the algorithm and a report (see details below).

Project Details

1. Read and understand your scientific publication.
2. Implement the method described in the publication in Python.
3. Measure performance of the algorithm: runtime, goodness metrics, performance metrics (see below).

The notion of “good” communities will be tested using the following **goodness metrics**:

- **Separability** captures the intuition that good communities are well-separated from the rest of the network.
- **Density** builds on the intuition that good communities are well connected.
- **Cohesiveness** characterizes the internal structure of the community. Intuitively, a good community should be internally well and evenly connected, i.e., it should be relatively hard to split a community into two sub communities.
- **Clustering coefficient** is a measure of the degree to which nodes in a graph tend to cluster together.

The conformance of the identified communities to the ground truth communities will be tested using the following **performance metrics**:

- **Precision** is the proportion of vertex pairs in the same identified community that are also in the same ground truth community
- **Recall** is the proportion of vertex pairs in the same ground truth community that are also in the same identified community.
- **F-measure** is the harmonic mean of precision and recall.
- **Normalized Mutual Information** considers all the possible community matching between the ground truth communities and the identified communities. For more details, see: http://arxiv.org/pdf/1110.2515v2.pdf
- **Rand Index** or simple matching coefficient, accounts for both the specificity and sensitivity of the identified communities. For more details, see: http://www3.nd.edu/~dial/papers/ASONAM11c.pdf

Required Reading

- *Defining and Evaluating Network Communities based on Ground-truth by J. Yang, J. Leskovec. IEEE International Conference On Data Mining (ICDM), 2012.*
- The slides under Resources on Piazza labeled: *Evaluating Community Detection for Networks with Ground Truth Information.*

Project Plan

1. The project duration is two weeks.
2. At the end of the first week, each team is required to submit parts (i) and (ii) of the “Submission Requirements.”
3. At the end of the second week, submit parts (iii) and (iv) of the “Submission Requirements.”

Submission Requirements

1. Algorithm code with detailed comments.
2. README file with detailed instructions. It should include the following information:

- Software that needs to be installed (if any) with URL's to download and instructions to install them.
- Environment variable settings (if any) and OS it should/could run on.
- Instructions on how to run the program, with specific run command example that can be copy pasted.
- Instructions on how to interpret the results.
- Sample input and output files.
- Citations to any software you may have used or any dataset you may have tested your code on.

In short, the TA should be able to install any required software, set up the environment, execute your program, and obtain results without any prior knowledge about your project.

3. **Project Report (2-3 pages)**
   This report should describe, in your own words, the algorithm discussed in the scientific publication assigned to you. This section should also discuss the performance of this algorithm. If the algorithm is parameterized, then include some discussion and empirical results on the effect of the parameters on the identified communities. Provide a detailed analysis of your algorithm and report the goodness metrics and performance metrics.

Project was created by Kanchana Padmanabhan and Nagiza Samatova