Implementation, evaluation and comparison of community detection algorithms

Theodoros Kyriakou
Department of Computer Science
University of Cyprus
Nicosia, Cyprus
tkyria06@cs.ucy.ac.cy

Andreas Panayiotou
Department of Computer Science
University of Cyprus
Nicosia, Cyprus
apanay20@cs.ucy.ac.cy

I. Abstract

In this paper we have implemented three different algorithms for Community Detection. These are the Label Propagation Algorithm, Girvan-Newman Algorithm and Louvain Algorithm. Then, we use some evaluation measures to see and compare the validity of the results of these algorithms. These measures are Mutual Information score, Normalised Mutual Information score, Rand Index score and Adjusted Rand Index score. Finally, we analyze and compare the data that have produced.

II. Problem Definition

Community detection in networks was a classic problem in graph theory for many years. Many algorithms and methods have been published to solve that problem. A community can be defined as a set of nodes that have more links within the set than outside it [9]. These links can be characterized by the content similarity between users, friendship between them and also other similarities in their personal data such as their location, gender, age etc. These close structures can then be used in many different areas for various purposes such as targeting marketing schemes, politics etc. In this work, we will study, implement and evaluate a list of the well known algorithms.

III. Problem Motivation

Technology, as time goes on, enters more and more into our lives and people interact with each other every day using the internet. These interactions can occur just about anywhere. Social Media, Emails and online shopping websites are a few examples. To be able to identify the communities through all of these interactions that happen online, allow us to understand how influence, knowledge, and even happiness, flow through a social network and it provides an approach to a real world problems [1]. This has motivated us to study and try to implement algorithms for Community Detection.

IV. Algorithms Description

Label Propagation Algorithm

Assume that a node x has neighbours x_1, x_2, ..., x_k. Each neighbor has a label that indicates the community they belong to. Afterwards, x decides its community based on its neighbors' labels. The community that each node in the network will have, is the community which most of its neighbours belong. We assign unique labels to each node and then let the labels spread across the network. As the labels spread, densely linked groups of nodes soon agree on a single label (see figure 1). When a large number of dense groups appear around the network, they continue to extend outwards until it is no longer possible. Nodes with the same labels are paired together as a community at the end of the propagation process. We perform this process repeatedly, where at every step, each node updates its label based on the labels of its neighbors. The order in which all the n nodes in the network are updated at each iteration is chosen randomly. It's important to remember that the algorithm starts with n distinct labels and gradually decreases over iterations, resulting in just as many unique labels as there are communities. Ideally, this operation will go on until no node in the network changes its label. Though, nodes in the network may have the same maximum number of neighbors in two or more communities. Because we break ties among the potential candidates at random, the labels on those nodes may change over iterations even though their neighbors' labels remain constant. As a result, we repeat the procedure until every node in the network has a label to which the maximum number of its neighbors belong to. Hence, the network is divided into disjoint communities, with each node having at least as many neighbors within its community as it does with any other. If C_1, ... C_p are the labels that are currently active in the network and $d_i^{C_j}$ is the number of neighbors node i has with nodes of label C_j, then the algorithm is stopped when for every node i,

$$\text{If i has label } C_m \text{ then } d_i^{C_m} \geq d_i^{C_j} \ \forall \text{ j}$$

At the end of the iterative process nodes with the same label are grouped together as communities. [2]

Pseudocode

```
1) Initialize the labels at all nodes in the network. For a
   given node x, $C_x(0) = x$.
2) Set t = 1.
3) Arrange the nodes in the network in a random order and
   set it to X.
```

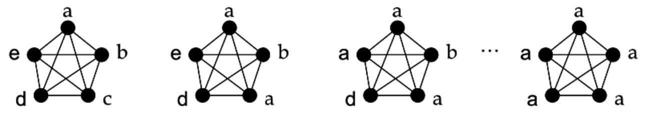

Fig. 1. Nodes are updated one by one as we move from left to right. Due to a high density of edges (highest possible in this case), all nodes acquire the same label. [2]

```
4) For each $x \in X$ chosen in that specific order, let $C_x(t) = f(C_{xi1}(t), C_{xim}(t), C_{xi(m+1)}(t-1), ..., C_{xik}(t-1))$.

   $f$ here returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly. If every node has a label that the maximum number of their neighbors have, then stop the algorithm. Else, set $t = t + 1$ and go to (3). [2]
```

Girvan Newman Algorithm

This algorithm, instead of attempting to develop a metric that tells us the edges are more central to communities, we concentrate on the edges that are least central. The edges that are most "between" communities. We build communities by slowly deleting edges from the original graph, rather than adding the strongest edges to an initially empty vertex set. The betweenness centrality of a vertex i is defined by Freeman as the number of shortest paths between pairs of other vertices that pass through i. It is a metric of a node's impact on the flow of information between other nodes. If there are several shortest paths between two vertices, each path is assigned equal weight so the total weight of all paths is unity. When a network includes communities or groups that are only loosely connected by a few inter-group edges, then the shortest paths between them must pass through one of these few edges. Thus, the edges connecting communities will have high edge betweenness. By removing these edges, we can isolate groups from one another and thereby expose the graph's underlying community structure. [3]

Pseudocode

```
1) Calculate the betweenness for all edges in the network.
2) Remove the edge with the highest betweenness. In case of tie, pick a random edge.
3) Recalculate betweennesses for all edges affected by the removal.
4) Repeat from step 2 until no edges remain. [3]
```

Louvain Algorithm

This algorithm is a fast, greedy algorithm which is based on the maximization of modularity. The initial algorithm can be applied both to unweighted and weighted graphs and consists of two phases that are repeated iteratively. First, the main goal of phase one, is to assign communities to the nodes until no improvements can be achieved. Specifically, the initial step is to assign a different community to every node of the network. Then, for each node i we calculate the modularity gain by moving it to the communities of every one of its neighbours nodes. Finally, we assign the community to the node i that produce the maximum gain in modularity, but only if this gain is positive. If no positive gain is possible, i stays in its original community. This phase is running repeatedly until no further improvement can be achieved. Note that, the sequence that we visit the nodes in phase one may affect the output of the algorithm. Another critical part of the algorithm, which improve its efficiency, is that to calculate the modularity gain is only necessary to calculate the modularity of moving a single node i in a single community C. This can be achieved using the formula below where gain in modularity is equal to:

$$\left[\frac{\sum_{in}+k_{i,in}}{2m}-\left(\frac{\sum_{tot}+k_i}{2m}\right)^2\right]-\left[\frac{\sum_{in}}{2m}-\left(\frac{\sum_{tot}}{2m}\right)^2-\left(\frac{k_i}{2m}\right)^2\right]$$

- $\sum_{in}$: sum of degrees of every node inside C,
- $\sum_{in}$: sum of the weights of the links inside C,
- k_i: sum of weights of the links incident to node i
- k_{in}: sum of weights of the links from i to nodes in C
- $m = \frac{1}{2}\sum_{ij} A_{ij}$

Now, considering the second phase of the algorithm, its main goal is to build a new graph whose nodes are now now the communities found during the first phase. Specifically, in each run of this phase, the graph is combined to a smaller one. This can be achieved by combining the edges, which both of their nodes are located inside the same community to a self-loop edge, with the sum of the weights, of a super node representing that community. Likewise, the edges connecting nodes from one community to nodes located into an other community combined to a single edge, having the sum of their weights, that connecting the super nodes of the two communities. At every run of the second phase, the algorithm is producing a new better partition for the graph. Finally, Louvain algorithm is simple and have many advantages, like its steps are easy to implement and also is very fast, something that make it ideal for applying in large scale networks. [4]

Pseudocode

(Consists of two phases)

Phase 1

```
1) Each node in the network is assigned to its own community.
2) For each node, the change in modularity is found by removing the node from its current community, and then moving it into the community of each neighbor of j.
3) Node is then moved to the community that resulted in the greatest increase of modularity. If no increase is possible, remains in its original community.
4) Repeat from step 2 until no change in modularity is possible.
```

Phase 2

1) Edges between nodes of the same community are now represented as self loops.
2) Edges from multiple nodes in the same community to a node in a different community are represented as weighted edges between the two communities.
3) Phase 1 is now run with this new smaller graph.
4) This is repeated until no gain in modularity is possible or graph combined to a single node. [4]

V. Evaluation Methods

Modularity

The modularity of the partition is widely used to determine the quality of partitions produced by these algorithms. It is a scalar value between -1 and 1 that compares the density of links within communities to the density of links among communities. Networks with a high value of modularity provide dense connections between nodes inside communities but sparse connections between nodes of different communities. Modularity can show whether a partition is good or bad, however is not absolute that the communities that produced are correct. Finally, the value of modularity is defined as follows: [4][5]

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

- A_{ij} : weight of the edge between i and j
- $k_i = \sum_j A_{ij}$: the sum of the weights of the edges attached to vertex i (degree of vertex i)
- c_i: the community to which vertex i is assigned
- $\delta(u, v)$: returns 1 if u = v and 0 otherwise
- $m = \frac{1}{2} \sum_{ij} A_{ij}$, the number of edges in the graph.

Mutual Information (MI)

The Mutual Information (MI) metric was established in the sense of classical clustering to compare two distinct partitions of a single data set by calculating the amount of information they share. This measurement is based on how much knowing one variable reduces uncertainty about the other. The Mutual Information score is a non-negative value and the higher the value, the more information is shared between two partition X and Y. [6] Finally, in order to evaluate the algorithms we have also used the normalized version of Mutual information, the NMI, which is bounded in the range 0 and 1, and again higher value means the partition that was produced is more similar to the ideal. [7]

$$MI(X, Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right)$$

- $p(x, y)$: probability of node x to exists in community X and node y to exists in community Y simultaneously
- $p(x)$: probability of node x to exists in community X
- $p(y)$: probability of node y to exists in community Y

Rand Index (RI)

The Rand Index measure is focused on pairwise agreement. It compares how two partitions handle each potential pair of nodes. The Rand Index score ranges between 0 and 1 and the higher the value is, the better is the output of the algorithm. Likewise, except from the Rand Index score we have used also the Adjusted Rand score which produce more accurate results in case the number of truth communities are not the same with the number of the communities that the algorithm produced. Rand Index score given by the formula below: [8]

$$RI(X, Y) = \frac{a + b}{a + b + c + d} = \frac{a + b}{\binom{n}{2}}$$

- n : number of nodes
- a : number of pairs which are in the same community in X and in the same in community Y
- b : number of pairs which are in different communities in X and in different communities in Y
- c : number of pairs which are in same community in X and in different communities in Y
- X number of pairs which are in different communities in P_i and in same community in Y

VI. Evaluation Results

In this section we will focus on the evaluation results. First, in case to be able to evaluate all three algorithms we had to find a dataset with known ground-truth communities, that all algorithms will be able to import and run. We decided to use the Zachary's Karate club graph, which is a social network of a university karate club, consists of 34 nodes, 78 edges and has 2 communities. Unfortunately, we were unable to use a larger network with ground-truth communities as first the Girvan Newman algorithm would be very slow, due to the calculation of betweenness centrality and second, our implementation of Louvain algorithm uses the graph adjacency matrix which could not be built because of limited computing resources. Thus, to evaluate the algorithms we run all of them a number of times in the network mentioned above, and calculated the average value from every evaluation measure. The results are shown below in Figure 2.

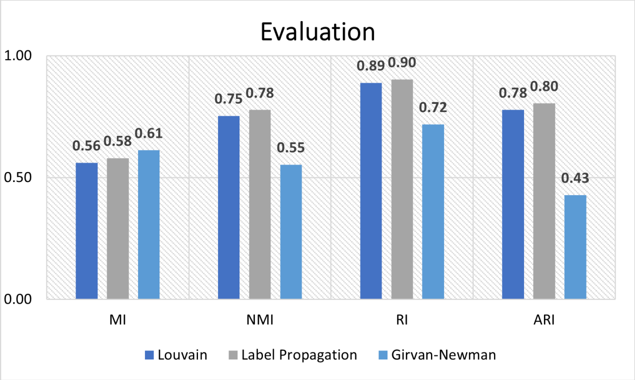

Fig. 2. Evaluation results for the three community detection algorithms and four evaluation measures mentioned.

As we can see the algorithms, in most cases, are producing partitions and extract communities that are similar to ground truth. Likewise, we can see that both Louvain and Label Propagation scores are very good, and compete each other, with the performance of Girvan-Newman implementation to be acceptable. Finally, considering Adjusted Rand Index, we see that Girvan-Newman implementation is scoring very low, comparing the other two implementations. This may happens because, in case of number of true and predicted communities are not the same, ARI score is affected negatively.

VII. Conclusion and Future Work

In this paper we had the chance to deal with community detection algorithms. As we already mentioned, is a very interesting topic in graph theory and also with many applications in real world problems. We managed to implement three community detection algorithms, the Louvain algorithm, the Label Propagation algorithm and Girvan-Newman algorithm. Then, using a list of evaluation measures and a dataset with known ground-truth communities, we saw how the algorithms perform. As a future work, would be very interesting not only to study and implement more algorithms that solve the same problem, but also to use them in real world data to extract useful information.

References

[1] Christopher Hogan, Community Detection Detailed for Online Social Networks, Wilfrid Laurier Univertsity, 2015.

[2] Usha Nandini Raghavan, Reka Albert, Soundar Kumara, Near linear time algorithm to detect community structures in large-scale networks, The Pennsylvania State University, 2007

[3] Michelle Girvan and M. E. J. Newman, Community structure in social and biological networks, 2001.

[4] Blondel V D, Guillaume J-L, Lambiotte R and Lefebvre E, Fast unfolding of communities in large networks, Universit´e catholique de Louvain, 2008.

[5] M. E. J. Newman, Analysis of weighted networks, University of Michigan, 2004.

[6] Cover, T.M.; Thomas, J.A. (1991). Elements of Information Theory (Wiley ed.). ISBN 978-0-471-24195-9.

[7] Strehl, Alexander, Ghosh, Joydeep: Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. Journal of Machine Learning Research, 2002

[8] W. M. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical Association, 1971

[9] M. Kewalramani, COMMUNITY DETECTION IN TWITTER, University of Maryland Baltimore County, 2011.