Large Scale Social Network Analysis

Deepthi Jallepalli, Lakshmi Vihita Kesiraju, Mohit Patel, Shravanthi Bhoopalam Sudharshan

Computer Engineering Department
San José State University (SJSU)
San José, CA, USA
Email:{deepthi.jallepalli, lakshmivihita.kesiraju, mohit.patel, shravanthi.bhoopalamsudharshan}@sjsu.edu

***Abstract*—Social Network Analysis is the study of measuring, mapping and analyzing the properties and patterns of various structural aspects of Network. It is useful for drawing the relational ties about different individuals as well as groups. One important use of social network analysis is visualizing complex networks. Social Network analysis is mainly classified as socio-centric networks or egocentric studies with a focus on understanding the patterns within the network. In egocentric networks focus on the relationship between ego and the named objects in the network. Such that the analyzed social structures can be applied for various purposes like data mining, network modeling and sampling, social filtering and sharing, developing recommendation systems, community-based problem solving, etc.**

***Index Terms*—Social Networks, Community Detection, Graphs, Ego-networks, Centrality**

I. Introduction

Social Network Analysis has grown massively in recent years. It is the study of structure, and how it influences health, and it is based on theoretical constructs of sociology and mathematical foundations of graph theory. Structure refers to the regularities in the patterning of relationships among individuals, groups and/or organizations.

Egocentric network designs, focus on a ego, and the relationships between the ego and named people or objects within their social networks. Egocentric study designs use either name generators or position generators to obtain both attribute and relational data that can be used to construct people-by-people from which egocentric data analysis can be constructed. Graphs are visual representations of a network.

In our analysis, we are studying the design of Facebook's egocentric network data of friends circles that focuses on identification of people or nodes who are most influential in the network using different metrics. We are also trying to understand the different communities present in the network using community detection algorithms and also predict link between two different people or nodes using like prediction methods. This gives us an inference about the entire egocentric network of Facebook's friends circles.

II. Motivation

The increase in the use of social networks and the amount of data being created from various networks given raise to need of analyzing and understanding of complex networks. This way of analysis is also useful for fraud detection, spam detection, sentiment analysis, recommendations and predictions.

Link prediction is useful to predict the changes in the network and make recommendations accordingly based on attribute information. Community detection is an imperative research area done by revealing the structure of network by segregating the different divisions to build better recommendations. Finding most influential user from a given network depending on the centrality of the nodes. Our project is designed to implement the discussed methods on a social network data to get a clear understanding and to analyze the networks in a better way.

III. Related work

Community detection being one of the developing areas in network analysis to study and analyze various networks. The main use of Label Propagation Algorithm is that it updates the labels of the nodes that do not belong to the same community creating unique communities by considering only the network information with linear time complexity[1]. However, LPA has issues with stability of recognized communities, however, it works in same linear time complexity.

Community Detection is applicable to every system whose data can be represented as graphs. For social networking where communities are related to study about the ego circles of a single user u and form a network of followers vi and we need to identify the circles to which each alter vi belongs. However there is huge overlap among these communities for which unsupervised clustering algorithms like Girvan-Newman and Louvain detect these overlapping communities and modularize the ego network into desired clusters[7].

Link prediction has been one of the most interesting issues in social network analysis. It exploits the existing information of the network, such as the features of the nodes and edges, to predict the formation of potential relationships in the future[8].

A lot of feature learning approaches are not expressive enough in capturing the diversity of connectivity patterns in a social network. For learning continuous feature representations for the nodes in the Facebook data, the algorithmic framework Node2vec outperforms the other feature generation techniques[9].

For finding suitable relays in message forwarding, understanding node sociality is of utmost importance. Online and offline community structures do not have significant correlation on most datasets. However, most of the offline strong

social ties correspond to online social ties. Also, in some cases, online and offline brokerage roles show high similarity[10].

IV. TECHNICAL APPROACH

A. Most influential user

To formally state the problem, we are considering an undirected social graph network G = (V,E) where V stands for the number of friends i.e., the nodes in the graph and edge, $(u, v) \in E$ where it represents the connection between user u and user v.

In complex network analysis, Centrality is a very important concept for identifying important nodes in a graph. It is used to measure the importance (or "centrality" as in how "central" a node is in the graph) of various nodes in a graph. Every given node could be important from an angle depending on how the metrics of "importance" is defined. NetworkX is a Python language package for exploration and analysis of complex networks. We have used networkX package to visualize the graphs.

a) Degree Centrality: Degree centrality is the most common and simplest centrality measure to compute. In an undirected graph, it is simply the count of the number of connections (i.e., edges) a specific node has. As discussed, every centrality measure can specify different type of importance, higher degree centrality, shows how many connections a person has. Higher the value, more influential that person is. In mathematical terms, The degree centrality for a node 'v' in the bipartite sets 'U' with 'n' nodes and 'V' with 'm' nodes is:

$$d_v = \frac{deg(v)}{m}, \text{for} v \in U, d_v = \frac{deg(v)}{n}, \text{for} v \in V,$$

where 'deg(v)' is the degree of node 'v'. Though simple, degree is often a highly effective measure of the influence or importance of a node: in many social settings people with more connections tend to have more power.

b) Betweenness Centrality: In graph mathematical terms, it would be the median of the data, i.e., the node which is in the middle. For computation of betweenness, for a node N, we select a pair of nodes and find all the shortest paths between those nodes. Then we compute the fraction of those shortest paths that include node N. We repeat this process for every pair of nodes in the network. We then add up the fractions we computed, and this is the betweenness centrality for node N. Betweenness centrality is given by:

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)}$$

where V is the set of nodes, $\sigma(s, t)$ is the number of shortest (s, t)-paths, and $\sigma(s, t|v)$ is the number of those paths passing through some node v other than s, t. If $s = t$, $\sigma(s, t) = 1$, and if $v \in s, t$, $\sigma(s, t|v) = 0$.

A person with greater value of betweenness may be followed by many others who don't follow the same people as the user. They can control the flow of information from one part to another.

c) Closeness Centrality: Closeness centrality looks for the node that is closest to all other nodes. It indicates how close a node is to all other nodes in the network. It is simply the calculated sum of the length of the shortest paths between the node and all other nodes in the graph. Mathematically, Closeness centrality of a node 'u' is the reciprocal of the sum of the shortest path distances from 'u' to all 'n-1' other nodes. Since the sum of distances depends on the number of nodes in the graph, closeness is normalized by the sum of minimum possible distances 'n-1'. It is given by:

$$C(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(v, u)},$$

where $'d(v, u)'$ is the shortest-path distance between 'v' and 'u', and 'n' is the number of nodes in the graph.

d) Eigenvector Centrality: Eigenvector/PageRank centrality is a measure to quantify the importance of nodes based on their influence for strategically connected nodes. The PageRank algorithm used by Google's search engine uses this centrality measure. The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. It is given by:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

where $p_1, p_2, ..., p_N p_1, p_2, ..., p_N$ are the pages under consideration, $M(p_i)M(p_i)$ is the set of pages that link to $p_i p_i, L(p_j)L(p_j)$ is the number of outbound links on page $p_j p_j$ and N is the total number of pages.

B. Community Detection

Discovering communities within billions of nodes of evolving network structures is an NP-hard problem . Our approach for community detection analysis is in-order to extract communities from undirected ego network, we must choose a scoring function (like centrality, betweenness, degree that we discussed in earlier in this section) that measures the intuition that communities relate to densely linked sets of nodes. After scoring function is decided we need to find sets of nodes with a high value of the scoring function. Following are the different types of algorithms we studied and implemented as part of community detection analysis.

a) Girvan-Newman Algorithm: The scoring function for Girvan-Newman algorithm is "edge-betweenness"(section IV b). The Girvan–Newman algorithm detects communities by recursively removing edges from the original network instead of creating communities from scratch like hierarchical clustering. The connected components of the remaining network are the resulted clusters.

In graph G =V,E, edge betweenness is an edge E that has the number of shortest paths between pairs of nodes that moves along it. Equal weights are assigned for all such shortest paths existing between a pair of nodes. There is possibility loosely connected different communities exists in given G are connected through at-least one among these

shortest paths that results in having high edge betweenness. By removing such edges connecting communities will have high edge betweenness (at least one of them). By removing these edges, the loosely connected groups are separated from one revealing communities existing in given graph.The algorithm takes input G and K = no.of clusters desired. Below steps describe the Girvan-Newman algorithm:

- Step 1: Compute the betweenness of all existing edges in given graph G.
- Step 2: Remove the edge(s) with the highest betweenness.
- Step 3: Re-compute the betweenness of all edges affected by the removal edge.
- Step 4: Steps 2 and 3 are repeated until no edges remain.

b) Louvain Algorithm: The scoring function for Louvain algorithm is "Modularity". It is believed that networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules. In given G with n nodes, initially, the algorithm considers every single node as individual cluster. For each node X, the algorithm there are Y neighbours considered. Now the two stages of the algorithm are applied for every X with Y neighbours belongs G.

- Process 1: Modularity Optimization: Compute the modularity of X and cluster X with each of it's Y neighbors. Recompute the modularity score changes the node is removed from its community and added to each of its neighbor's communities. The node is then positioned into the community where the gain in modularity is maximized. This is recursive process for each of the nodes up until no further improvement can be achieved.
- Process 2: Community Aggregation: In this process, the algorithm constructs new network whose nodes are now the communities found in the first process. Until there are no more changes to be made and a maximum of modularity is achieved these two-process run iteratively.

c) Label Propagation Algorithm: It is an algorithm that does not require any prior knowledge about the objective of the function but only the network structure. It estimates a single label in a densely connected graphs and different community oriented spaces influencing the generating of communities in a dissimilar pattern. The following steps is the approach to achieve community detection using Label Propagation.

• Initialize all the nodes in a network with a sample label (say x). Cx(0)=x.

• Re-arrange all the nodes in a random order of a network, represented as X.

• For every node in the network X, it returns the high frequency occurring label among its neighbors. • If there are more than one label with highest frequency, selects one randomly.

• The algorithm is stopped, if every node in the network got a label with highest occurring frequency.

Among the different implementations of LPA, Synchronous Label Propagation Algorithm is easy to parallelise and is also stable, but the labels may oscillate for each of the iteration making the stop criterion to be difficult. Asynchronous Label Propagation Algorithm, overcomes the problem of back and forth label oscillation by sequentially updating the vertices based on existing labels[4]. Although, it has also got few disadvantages such as instability of community count, creating monster communities and dependencies to be considered.

d) Evaluating Detected Communities: After successfully discovering communities using above descried community detection algorithms we need to evaluate how well is the clustering result. We followed two different strategies to evaluate the predicted clusters.

- Evaluate with respect to ground truth table: We incorporated a new way of evaluating clusters by computing Loss/Cost score using predicted and actual circle[6]. Where the Loss/Cost score is defined as minimum no.of edits required to adjust in predicted circles to make it up to true circles. In the dataset we are provided with circles files that have true communities for each ego network.
  Consider an ego graph G = (V,E) , A = set(predicted communities) B = set(True communities).
  1. Compute m*m matrix where m = size of the set of larger community set and find no.of edits for each community in predicted circles vs each community where.
  2. No.of edits = $A \cup B - A \cap B$
  3. Finally convert the matrix into cost matrix,cost[][] and a position (m, n) in cost[][],final score is sum of all minimum costs on that path to reach (m, n) from (0, 0).Lower the loss/cost score is better the algorithm.

- Evaluate without ground truth table: In this process, we evaluate the goodness of each cluster that based on few parameters like Separability, Density, Cluster Coefficient[5] that should have higher score to prove goodness of cluster.
  For each ego network G assume S is community of n nodes, m edges) Separability is computed as the ratio between the internal and the external number of edges:

$$Separability = \frac{m}{E-m}, \text{for} E \in G$$

  Density is computed as the fraction of the edges (out of all possible edges) that appear between the nodes in S:

$$Density = \frac{m}{n(n-1)/2}$$

  Cluster Coefficient is average of local clustering coefficients C which is the degree to which nodes in a graph tend to cluster together:

$$ClusterCoefficient = \frac{1}{n-1}\sum_{i} C$$

$$C = \frac{P}{k(k-1)}$$

  where Pi is proportion of links between the vertices within its neighbourhood , K is possible links.

C. Link Prediction

In this section, we propose a model to predict if a pair of unconnected nodes will form a link in the future.

1) Data Preparation: NetworkX library was used to plot a graph for our data. Using NetworkX's nodes function we identified all the nodes in our data and added them to a list. Using this list and the list of edges from our data we identified all the unconnected edges in the network and added them to a dataframe. The dataframe had two columns, one for each node in a pair. Each row in the dataframe represented a pair of unconnected nodes. We added a third column to the dataframe and set all of its values to 0. 0 in the column indicated that there was not a link between the two nodes whereas 1 meant that there was a link between the two nodes. We then moved all the omissible edges from our data to the dataframe and set all of its corresponding column values to 1. An omissible edge is one that can be removed from the graph without producing an isolated node.

2) Feature Generation: For feeding data to our link prediction model, the data had to have some features. There were several techniques available for extracting features from the nodes of a network. The Node2vec framework outperformed the other algorithms when applied to the Facebook dataset. Using the data obtained after removing the omissible edges, we generated walks and trained the Node2vec model to compute the features of every node in the network.

3) Model Selection: In order to predict links in our data we used sklearn's LogisticRegression model. Logistic Regression proved useful since we were trying to explain a relation between a dependent binary attribute and one or more independent nominal, ordinal, interval or ratio type attributes. In our case, the binary attribute was the column that contained values- 0 or 1; 0 meaning no link between the nodes and 1 meaning that there was a link between the nodes. The logistic model was the best choice to compute the probability of link formation in our data.

4) Predicting Links: To predict links in our network we fed the vectors generated from the Node2vec algorithm to train sklearn's LogisticRegression model. Then using LogisticRegression's predict function we computed the probability of link formation for all the unconnected pairs. To minimize false recommendations, we set the probability threshold to 0.9, which meant that only the nodes with a probability of forming a link higher than 0.9 were given as output.

V. Experiments and Results

A. Dataset

This project uses Facebook dataset from SNAP large network projects that is available at http://snap.stanford.edu/data/ego-Facebook.html. This data was collected from survey participants using this Facebook app. The complete Facebook ego network is undirected graph that has 10 ego networks, 4039 nodes, 88234 edges and Average degree is 43.6910. Where for each ego network we have 5 different files namely edges, circles, feat, egofeat and featnames.

- nodeId.edges : Contains all edges in the ego network for the node 'nodeId'.
- nodeId.circles : Contains the set of circles for the ego node. Each line contains one circle, consisting of a series of node ids.
- nodeId.feat : Contains features for each of the nodes that appears in the edge file.
- nodeId.egofeat : Contains features for the ego user.
- nodeId.featnames : Contains names of each of the feature attribute where the values for attribute is 1 if followed by user else 0.

B. Centrality Measures

The EgoNode 689 is selected for our analysis which has 62 nodes and 331 edges. The average degree for this EgoNode is 10.6774.

a) Degree Centrality: In this algorithm, we take the graph and its nodes as input. The degree centrality values are normalized by dividing by the maximum possible degree in a simple graph n-1 where n is the number of nodes in G. A plot where the node color varies with Degree and node size with Degree Centrality is shown in figure[1].

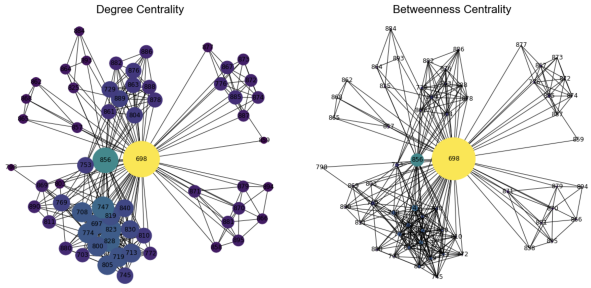

Fig. 1. Degree Centrality and Betweenness Centrality

b) Betweenness Centrality: In this algorithm, the graph G is given as an input and K is chosen as node samples to estimate betweenness. The value of k ¡= n where n is the number of nodes in the graph. Higher values give better approximation. We normalize the measure by $2/((n-1)(n-2))$ for graphs, and $1/((n-1)(n-2))$ for directed graphs where n is the number of nodes in G. number of undirected paths when G is undirected. A plot where the node color varies with Degree and node size with Betweenness Centrality is shown in figure[1].

c) Closeness Centrality: In this algorithm, the graph G is given as an input and using specified edge attribute as the edge distance in shortest path, closeness centrality is calculated. If the graph is not completely connected, this algorithm computes the closeness centrality for each connected part separately scaled by that parts size. A plot where the node color varies with Degree and node size with Closeness Centrality is shown in figure[2].

d) PageRank: In this algorithm, The PageRank computations require several passes, called "iterations", through the collection to adjust approximate PageRank

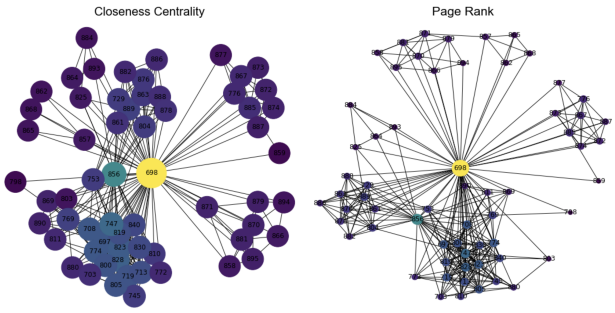

Fig. 2. Closeness Centrality and PageRank

values to more closely reflect the theoretical true value. The eigenvector calculation is done by the power iteration method and has no guarantee of convergence. A plot where the node color varies with Degree and node size with PageRank is shown in figure[2].

Comparative analysis of all the centrality metrics for the top 10 nodes in the graph is plotted to analyze the metrics. A comparative analysis shows that all the centrality measures for top 10 nodes range between similar values. The results are shown in figure[3].

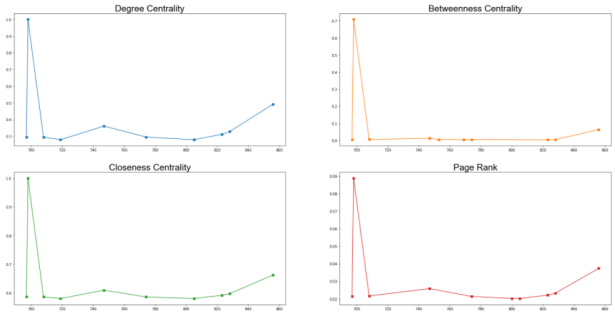

Fig. 3. Comparative Analysis

C. Community Detection

a) Girvan-Newman Algorithm: For each ego network, the edges and nodes are extracted from data files and using NetworkX, undirected graph is constructed that representing ego network. We implemented Girvan-Newman algorithm without any input parameter K which resulted in bipartion of the given graph. Also we run Girvan-Newman for K = 5 clusters as input and retrieved 5 communities for the same ego network. For Girvan-Newman we can control the K factor also, the only betweennesses being recalculated are only the ones which are affected by the removal, which improves the time complexity of the algorithm. Girvan-Newman algorithm can detect smaller and larger communities successfully. Girvan-Newman algorithm time complexity is O(mn) where m is the number of edges and n is the number of vertices.The discovered communities are differentiate using different color codes for each community.

b) Louvain Algorithm: Using same ego nodes and edges of each ego network we ran Louvain Algorithm to observe how does modularity optimization help in discovering communities.For this algorithm we cannot control the factor K, the no.of clusters to be identified. The algorithm detects the smaller communities and then merges them to bigger community that automatically discovers n no.of clusters. There is high possibility the Louvain algorithm might not have smaller communities in the result set.Time complexity for this algorithm is O(nlog2n) which is comparatively faster than Girvan-Newman with K = 5.The discovered communities is represented using different color code. The plots of evaluation results in shown in figure[4]. The plots depicting the evaluation

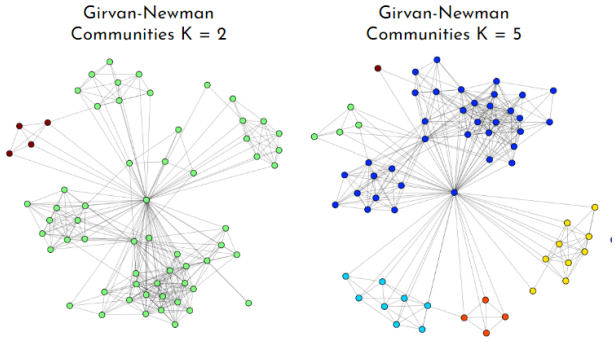

Fig. 4. Community detection using Girvan-Newman and Louvain

results are shown in figure[5].

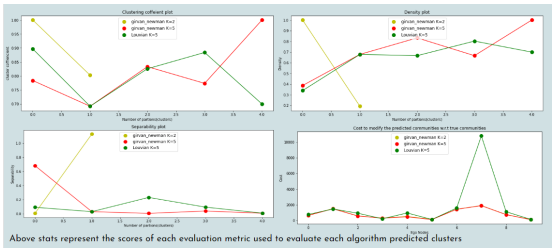

Fig. 5. Evaluation Metrics for Girvan-Newman and Louvain

c) Label Propagation Algorithm: We have implemented Community detection using Label Propagation Algorithm on the snap data set that has undirected data for Facebook social network for different users represented as ego networks. The following is the community separation for one the ego networks with each color nodes representing each community.

Although LPA being one of the fastest algorithm, there are few components to be kept in mind while evaluating its performance such as stability, quality, convergence, one-hop horizon and community count. The following are the graphical representations of evaluation metrics of Separability, Density, Clustering Coefficient and Cost to modify. figure[7]. The table in figure[8] shows detailed comparisons of all community detection algorithms implemented.

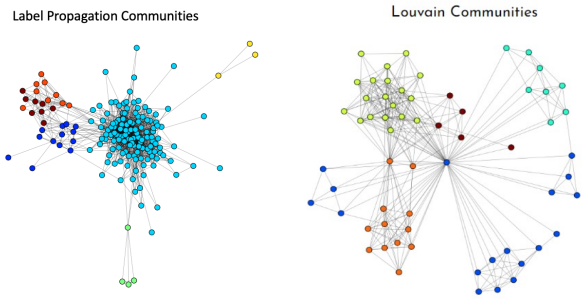

Fig. 6. Communities detected by Label Propagation and Louvain

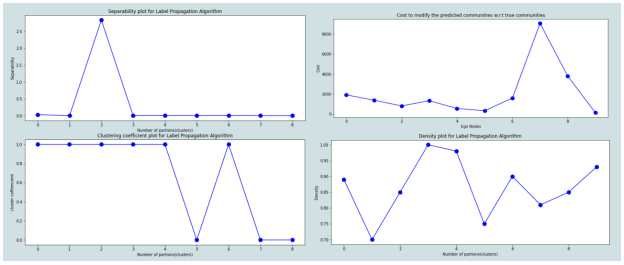

Fig. 7. Evaluation Metrics for Label Propagation

Parameters	Girvan-Newman k=2 / k = 5	Louvain K = 5	Label Propagation K =9
Approach	Partitions of given graph based on edge betweenness	Hierarchical clustering based on Modularity	Partitional algorithm based on epidemic spreading.
Avg Time taken	0.0013 secs / 0.0015 secs	0.458 secs	0.059 secs
Separability[Ego698]	0.570 / 0.155	0.093	2.882
Density[Ego 698]	0.596 / 0.713	0.637	0.619
Cluster Coefficient[Ego 698]	0.901 / 0.816	0.799	0.973
Modularity	0.005 / 0.103	0.457	0.510
Average Loss	807.3 / 790.4	1821.3	2083.3

Fig. 8. Comparison Analysis for Community detection Algorithms

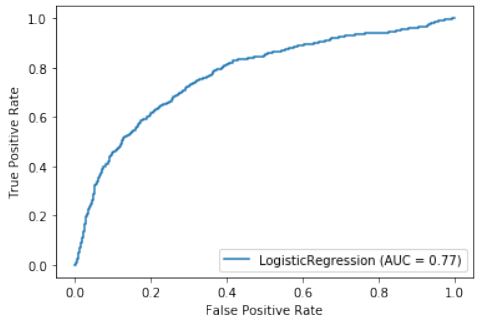

Fig. 9. Receiver Operating Characteristics curve for Link Prediction model

D. Link Prediction

The EgoNode 686 contained 168 nodes and 1656 edges. The EgoNode had an average degree of 19.71. The LogisticRegression model was trained using the data prepared from this EgoNode. The probability of link formation between the unconnected edges in the EgoNode 686 was then computed using this Logistic model. All unconnected node pairs with a probability of link formation greater than 0.9 were identified and given as output.

For evaluating our Logistic Regression model, we split the data into training and testing sets in the ratio of 7:3. We used the Area Under the Receiver Operating Characteristics Curve as evaluation metric. The reason behind selecting this evaluation metric was its ability to determine optimal cutoff values based on what we thought was a clinically appropriate trade off between the true positive rate and the false positive rate. On performing the evaluation on EgoNode 686 we obtained a score of 0.77[9]. However, the score falls to 0.72 when it is performed on the entire dataset, which is still a fair score for a link prediction model.

VI. Conclusion

We successfully implemented various algorithms and techniques to analyze large scale social networks like Facebook. We could predict the most influential user in the cluster for nodes. Evaluating communities is challenging as we observed the number of clusters predicted vs true are not same always so generic clustering evaluating metrics like F1 Score, Normalized Mutual Information(NMI) score, Accuracy score didn't fit to evaluate our community detection algorithms. Using our Link Prediction model, we could successfully predict links between unconnected nodes.

Acknowledgment

We thank Dr. Wencen Wu for guidance and lecturing us on Link Analysis and Community Detection in Graphs that made a smooth knowledge transition in this project. Also we are thank team for this collaborative effort and learning from each other.

References

[1] XIA Lei, ZHANG Lejun, ZHANG Jianpei, YANG Jing, GUO Lin, An improved Community Detection Algorithm based on Local Information in Social Networks, International Conference on Computational Intelligence and Security
[2] Sara E. Garza, Satu Elisa Schaeffer, Community detection with the Label Propagation Algorithm: A survey, Science Direct
[3] Xuegang Hu, Wei He, Role-based Label Propagation Algorithm for Community Detection, School of Computer and Information, Hefei University of Technology.
[4] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Santa Fe Institute, Santa Fe, NM
[5] Jaewon Yang, Jure Leskovec, Defining and Evaluating Network Communities based on Ground-truth.
[6] Kaggle competition "Learning Social Circles in Networks" community detection evaluation solution.
[7] Julian McAuley, Jure Leskovec "Learning to Discover Social Circles in Ego Networks".
[8] Tingli Wang, Guoqiong Liao "A Review of Link Prediction in Social Networks".
[9] Aditya Grover, Jure Leskovec "node2vec: Scalable Feature Learning for Networks".
[10] A. Socievole, F. De Rango, A. Caputo "Opportunistic mobile social networks: From mobility and Facebook friendships to structural analysis of user social behavior".
[11] GitHub Repository : https://github.com/Deepthi-Techhub/CMPE-256-Project-Repo