Fast and Approximate Network analysis using Graph Neural Networks (GNNs)

Nidhin Harilal
17110092
IIT Gandhinagar

Dyavarashetty Peeyush
17110053
IIT Gandhinagar

Kakumani Prudhvi
17110056
IIT Gandhinagar

Abstract

In network analysis, the problem of finding influential nodes has high theoretical and practical significance. Centrality measures are common schemes that aims to find these influential nodes in a network. However, deterministically computing these values for individual node takes a lot of time and are computationally expensive. This problem elevates when the analysis is done for larger graphs. Getting an approximate estimates of these centrality measures with much lower time would be of great significance in network analysis. In this project, we aim to do the same using Graph Neural Networks (GNNs). We train a GNN based model on a synthetically generated dataset consisting of variety of networks to approximate these centrality values. Our tests on complex real networks datasets shows that GNN gives promising results in predicting these values along with a huge speed-up of nearly 80-times when compared to the currently used deterministic algorithms.

All the codes, processed datasets, and trained models are in public domain[1].

1 Introduction

The problem of identifying the important nodes and ranking it in a graph is very significant in the field of network analysis. Here, influential nodes refers to certain nodes that can affect network structure and function to a greater extent than other nodes in the network. One of the ways in which influential nodes can be identified is ranking is based on its ability to control the spread of information in the graph [6]. Other measures include the degree of connectivity of nodes [8] and the closeness of the nodes in a network [7] etc. For the scope of this project We will be focusing on Betweenness , Closeness and Degree centrality.

1. **Degree centrality:** It refers to the number of connections that a node v has with the other nodes.
$$C_D(v) = deg(v)$$
2. **Closeness centrality** It refers to the inverse of the average length of shortest path between the node v and all others.
$$C(x) = \frac{N}{\sum_y d(y,x)}$$
3. **Betweenness centrality** It refers to the number of times the node v acts as a bridge in the shortest path between any of the two nodes.
$$C_B(v) = \sum_{s\neq v\neq t\in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

[1]`https://github.com/cryptonymous9/Network-Centrality-using-Graph-Neural-Nets`

The traditional method of computing these centrality values takes a lot of time and are computationally expensive. This situation worsens when the network size increases. One of way of solving this issue is by computing approximating values using Graph Neural Network (GNN) [5].

Graph Neural Network (GNN) is a type of Neural Network which operates directly on graphs. Originally,Graph Neural Networks were introduced back in 2005 but it regained attention in the past few years. Studies [9] [2] have shown that Graph Neural networks have a great representation power and have been successfull in establishing relationship between large scale graph and its features. Although there exists many approaches towards calculating approximate centrality measures, but complex and non-linear dependence among these values motivates the usage of Graph Neural Networks (GNN).

2 Data

Training Graph neural network for predicting centrality values will require a dataset which consists of networks with varies number of nodes, edges and which have diverse properties. For this, we have generated a synthetic dataset of consisting of 450 graphs for which included network types like Erdős–Rényi(ER), Barabási–Albert(BA) and Gaussian Random Partition graphs, each sharing atmost one-third of the total. The definition of each of these types are as follows:

- **Erdős–Rényi (ER):** Random graphs in which edges are set between nodes with equal probabilities.
- **Barabási–Albert (BA):** Random graphs having preferential attachments having a heavy tailored degree distribution.
- **Gaussian Random Partition:** Random graphs created using k-partitions each with a size drawn from a normal distribution.

Each of the graphs have number of nodes which have been samppled randomly from $2,000$ to $15,000$. These different graph types and the statistics of nodes and edges ensures diversity in the dataset. These graphs are built in python using 'networkx', 'networkit' and 'SNAP' scientific libraries. As a part of validation of the model, We collected data from Stanford Network Analysis Project(SNAP) Repository[2] and computed their centrality measure using python libraries. The collected validation data includes survey from participants in Facebook[3], Theory collaborations in High Energy Physics and collaborations in General Relativity and Quantum cosmology[4]. **Table ??** shows the statistics of the downloaded dataset.

Graph	No of Nodes	No of Edges
Facebook Survey	4039	88234
Physics Theory	9877	25998
General Relativity	5242	14496

Table 1: Graphs for test data

3 Methodology

Graph Neural Networks (GNN) have the ability to extract and learn complex relations in High-dimensional graphs as shown by [9], [2]. Most of the recent work that is done in GNN utilizes the idea of message passing framework. This involves the aggregation of node features at different levels from it's neighbours.

As explained in [5], Consider $G = (V, E)$ to be the graph consisting of node feature vetors. As shown in **Figure 1**, at each layer of the Graph Neural Network (GNN), the node feature vectors are updated by first aggregating features from neighbors and then adding it to its

[2]`http://snap.stanford.edu/data`

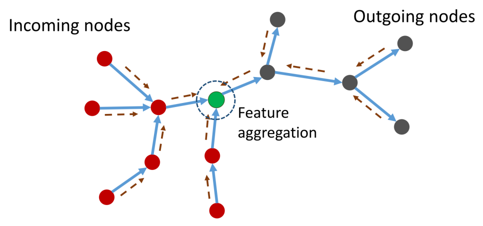

Figure 1: Message passing framework in Graph Neural Networks [5]

own feature vector. Mathematically, it can be formulated as [5]:

$$a_v^{(k)} = AGG^{(k)}(h_u^{k-1}), \text{ where } u \in N$$

$$h_v^{(k)} = COMB^{(k)}(h_v^{k-1}, a_v^{k-1})$$

where AGG refers to aggregtaion of neoghbour features of node v at the k the layer and $COMB$ refers to combining the aggreageted features with its own features.

3.1 Model Description

The architecture that we have used follows the same idea and accumulates the feature vectors at different levels of GNN layers that are attached consecutively to form a output vector as shown in **Figure 2**.

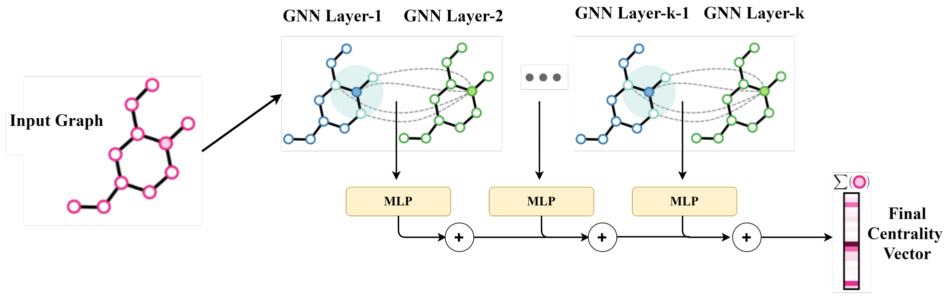

Figure 2: Model Diagram [5]

This Graph Neural network model is initialized with a learnable embedding lookup weight matrix. The MLP blocks in **Figure 2** refers to *Multi-Layer Perceptron*, acts as a *encoder-decoder* taking the node features from the GNN layers and building a latent-vector that would help in building up the final output vector. Each such latent-vectors are aggregated from the described diferent levels of the GNN hierarchy, which is corresponds to the idea of message passing.

3.2 Loss Criterion

Centrality values of a network are ordinal in nature making them ordinal variables. In machine learning ordinal variables refers to those which have a relative ordering between them that is of high significance. This makes the problem of predicting the centrality values as an ordinal regression problem. Therefore, standard regression criterion such as *Mean-Square Error* would be not be good objective functions. Hence, we have used the standard ranking loss, i.e Margin-Ranking loss (MR Loss) [1]as the objective function for training.

$$\text{MR Loss}(x, y) = max(0, -y \times (S_i^{pred} - S_i^{actual}))$$

$$\text{where } y = \begin{cases} 1 & \text{if } S_i^{pred} \geq S_i^{actual} \text{ in rank} \\ -1 & \text{otherwise} \end{cases}$$

3.3 Experimental settings

We have used a total of 6-stacked GNN layers and each of the MLPs(*Multi-Layer Perceptrons*) had a depth of 3 and each having 32 neurons respectively. Each GNN layer has a *dropout-rate* of 0.3. The model was trained for a total of 20 epochs, out of which the configuration with best test accuracy is considered for validation. The model configuration are same for all the three cases: Degree, Betweenness, Closeness centrality except for the dropout rates which is higher for the case of Closeness with 0.5. The model is built completely on PyTorch framework. For training NVIDIA-RTX 2080-Ti (12 GB) GPU along with an Intel Xeon Silver (x48 cores) processor was harnessed.

4 Results

As a part of evaluataion, we have considered both the accuracy in predicting the centrality values and the speed-up in time between our approach and the deterministic algorithm.

4.1 Prediction Accuracy

In network analysis, since we are concerned about the influential nodes when we compute centrality measures, therefore, we use the hit rates as a metric for performance of the model. The measure of $k-$hit rate here refers to the percentage of the nodes that the model retrieved/ predicted corectly in the top k percent influential nodes having the highest centrality values.

Centrality Measure	Top 5% Hits	Top 10% Hits	Top 20% Hits
Degree	89.05 ± 1.57	93.14 ± 1.20	99.11 ± 0.68
Betweenness	83.20 ± 1.64	90.58 ± 1.47	98.25± 2.40
Closeness	78.92 ± 4.16	85.07 ± 3.84	89.25±4.62

Table 2: Obtained Hit Rates along with its standard deviation on the test data.

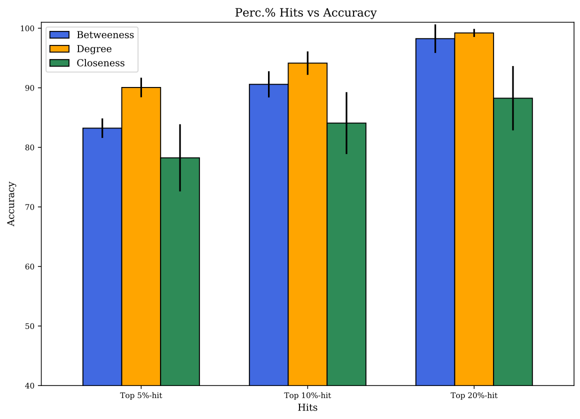

Figure 3: Plot of the obtained Hits vs Accuracy on the test data.

Table 2 and **Figure 3** shows the obtained percent hit rates on the test data. We were able to predict the degree and betweenness centrality measure with a good accuracy of more than 90 percent on top-10% hits. Closeness centrality on the other hand have a little lower but fair accuracy of more than 85% in top-10% hits in the actual network.

4.2 Time Comparison

Inference in Graph Neural Network (GNN) have a time complexity of order $O(E)$, where E refers to the edges in the graph. Solving both *Betweenness* and *Closeness* centrality measures deterministic ally take time in the order ON^3, where N is the number of nodes of the network. Therefore, GNN-based model should theoretically take much lower time than the deterministic algorithm. *Degree* centrality on the other hand, has time complexity of order $O(E)$, which is same as that of GNN-based model. However, due to higher number of computations in the GNN-based approach, it should take a loger time.

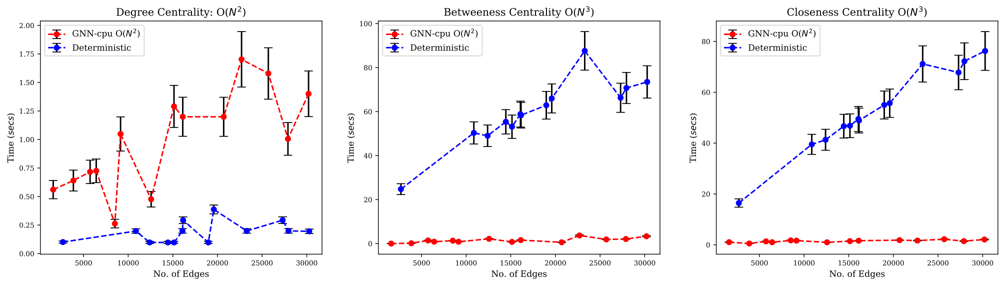

Figure 4: Comparison of the time taken in CPU to compute centrality measures, when the number of edges are increased in a network. The black bars show the std-deviation in time taken.

Figure 4 shows the comparison of the time taken by both GNN-based model and the deterministic algorithm to compute each of the centrality values as the number of edges are increased in a network. For a fair comparison in terms of hardware, the comparison shown in **Figure 4** is performed entirely on CPU. These results match our theoritical expectation. Note here the speed-up that GNN-based model offers when *Betweenness* and *Closeness* centrality are computed. For around $30,000$ edges, the GNN-based model offers almost 80 times speed-up which is tremendous considering the fact that it gives more around 90% accuracy on top-10% nodes in the network.

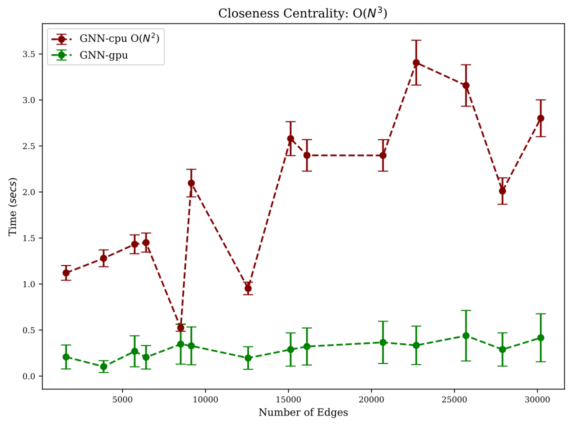

Figure 5: Comparison of time taken for computing *Closeness* centrality on GPU (NVIDIA-2080Ti) and CPU (Intel Xeon Silver).

Still, One of the other main feature of using GNNs are that frameworks such PyTorch and Tensorflow enables its computation over GPUs which should be theoretically much faster as compared to the computations over CPU. **Figure 5** shows the time comparison while computing on GPU and CPU. The plot shows that we can obtain further speed-ups when the computations are done over GPU.

5 Conclusion

The huge time speed-up that is provided by the Graph Neural Network (GNN) based model in computing centrality values justifies the advantage of exploring this field. GNNs are a very active line of research and in this project, we have used a simpler version of GNN-architecture which still performed fairly-well. This further motivates applying the recently proposed and far advanced GNN-based models for computing centrality values on the network. This whole approach can also be extended further to other Network measures like Page-Rank and clustering coefficients etc, which again are computationally very expensive to compute. On the limitation side, GNN still cannot take graphs which have higher nodes as compared to its model-size. Thereofore, work still needs to be done for generality of GNN for larger networks.

All the codes and the data used are present in public domain[3]

References

[1] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. pages 315–323, 01 2009.

[2] Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. In *NeurIPS*, 2019.

[3] Jure Leskovec and Julian J. Mcauley. Learning to discover social circles in ego networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 539–547. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4532-learning-to-discover-social-circles-in-ego-networks.pdf`.

[4] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

[5] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Fast approximations of betweenness centrality with graph neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2149–2152, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3358080. URL `https://doi.org/10.1145/3357384.3358080`.

[6] M.E. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27:39–54, 10 2003. doi: 10.1016/j.socnet.2004.11.009.

[7] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. Ranking of closeness centrality for large-scale social networks. volume 5059, pages 186–195, 06 2008. doi: 10.1007/978-3-540-69311-6_21.

[8] Akrati Saxena, Vaibhav Malik, and Sudarshan Iyengar. Estimating the degree centrality ranking of a node. *ArXiv*, abs/1511.05732, 2015.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2020.

[3] `https://github.com/cryptonymous9/Network-Centrality-using-Graph-Neural-Nets`