MIS 548 Final Report - Community Mapping with GNN

Group 11
Kiwoon Hong, Rishabh Bhonsle

1. Business Problem

1.1 Introduction

Social networks offer valuable insights into human interactions. They not only facilitate human relationships regardless of physical distance, also can be used as a platform for real time knowledge/information sharing. Also, social networks group people with like interests, build connections, and form communities. Although social networks have these advantages, they also come with drawbacks. One of them is the spread of fake news and misinformation. Information dissemination through social networks is real-time, which poses limitations on verifying the authenticity of the information.

Given the significance of social networks as elucidated above, they are well-suited for implementing GNN (Graph Neural Networks). In other words, the GNN model may be the best model for social networks. The architecture of GNN, comprising nodes and edges, bears similarity to the structure of social networks. Social networks can be represented as graph structures, with individual pages serving as nodes and the connections between pages represented as edges.

Hence, in this project, we will explore GNN models using social network data, focusing particularly on SNAP's Facebook Large Page-Page Network data. Our aim is to identify, understand, and classify social communities within this dataset. Additionally, leveraging insights gained from classification, our goal is to derive valuable business insights from the analysis of social networks.

1.2 Business Applications

The expected applications can be divided into three aspects: Accurate marketing strategies, Improving user experiences and augmenting customer segmentation. First, with accurate page classification, we can offer more sophisticated targeted marketing to customers, thereby improving advertising efficiency, including marketing cost efficiency and marketing accuracy. Second, by offering users content recommendations tailored to their communities, we enhance user experiences. This fosters greater consumer trust in the platform, subsequently resulting in heightened platform loyalty. Finally, we can define our target audience groups, in other words, customer segmentation, making it easy to understand and manage the customer in a group.

1.3 Related Work

We reviewed related works that use multinomial classification using GNN models for social networks and other domains. Li et al. proposed a novel end-to-end multi-omics GNN framework for accurate and robust cancer subtype classification. The proposed model integrates learned graph features and global genomic features for accurate classification. Awasthi et al. utilized a Double-Layered Graph Convolution layer model. The dataset used was the Feather-Deezer-social dataset from the Stanford Large Network Dataset Collection. They highlighted an example of predicting the gender of a user in the Deezer app, where the model achieved an impressive accuracy of nearly 88% on a test set. Overall, this study demonstrated the effectiveness of GNNs in the task of node classification and prediction in social networks.

2. Project Goals

Goal: To build a multinomial classification model to accurately classify the category/community of the facebook page under consideration.
Explanation: Social network data is inherently characterized by complex relationships and correlated features. Traditional classification techniques applied to such data have limited performance because simple feature vectors cannot fully reflect the complex connections between nodes.
To overcome these limitations, we propose an approach that leverages the inherent graph structure of social networks. Specifically, we use graph neural network (GNN) models to reflect inter-node connection information directly into the node representation to improve classification performance.
In short, GNN-based approaches can effectively leverage the graph-structured nature of social network data to improve classification and prediction performance. By internalizing complex node relationships into feature vectors, you can create more accurate and insightful models.

3. Data

3.1 Data Source

We utilized SNAP(Stanford Network Analysis Project)'s Facebook Large Page-Page Network data. The data was collected in November 2017 using the Facebook Graph API as an undirected webgraph. The data and embedding was proposed by Rozemberczki et al. The paper discusses Multi-Scale Attributed Node Embedding, a novel approach to network embedding algorithms that capture information about nodes from the local distribution over node attributes[4]. They utilized social networks and web graphs that they collected (e.g Twitch, Facebook, Github, Wikipedia) because social networks exhibit diverse types of relationships and interactions between nodes.

3.2 Data Description

Nodes correspond to official Facebook pages, links to mutual likes between sites. Node features are extracted from the site descriptions and encoded into numbers. The data table we'll use contains the id, name, target label (type of page), and features and edge values for each page. Edge values represent each index of edges.

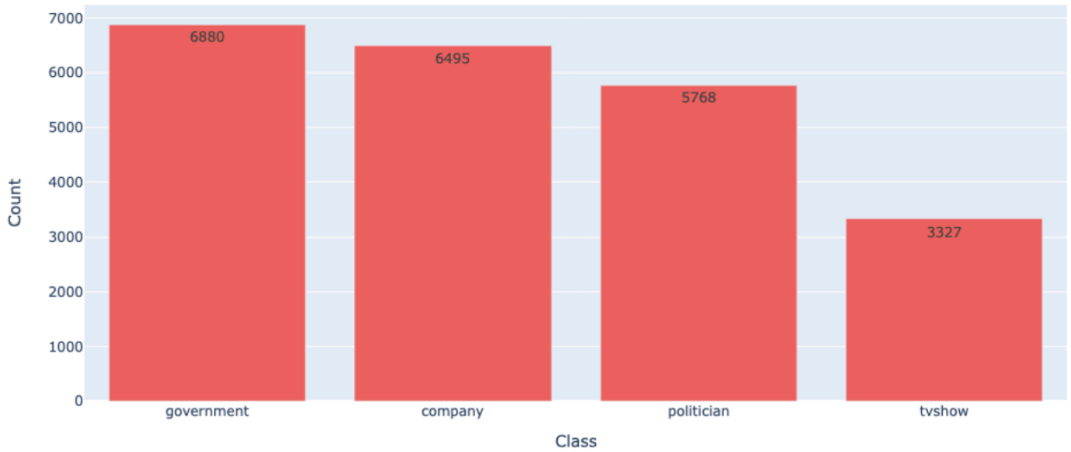
	id	facebook_id	page_name	page_type	features	edge_values
0	0	145647315578475	The Voice of China 中国好声音	tvshow	[3133, 3825, 236, 874, 1072, 143, 1078, 901]	[18427]
1	1	191483281412	U.S. Consulate General Mumbai	government	[3399, 597, 979, 2014]	[21708, 22208, 22171, 6829, 16590, 20135, 8894...
2	2	144761358898518	ESET	company	[3383, 3832, 2035, 765, 3972, 3364, 663, 2163,...	[9048, 6353, 2629, 11537, 13205, 22304, 17728,...
3	3	568700043198473	Consulate General of Switzerland in Montreal	government	[2710, 1960, 1940, 4514, 4339, 761, 2263, 1340]	[16742, 293, 5826, 3479, 19753, 17346, 10945, ...
4	4	1408935539376139	Mark Bailey MP - Labor for Miller	politician	[2873, 4518, 4535, 1602, 3500, 4457, 1910]	[13645, 20876, 11446, 16203, 2830, 2004, 20624...

3.3 Statistics

The data has 22,472 nodes and 171,003 edges. Each node has a different number of edges. Also, there are 4,714 features in the data. The average degree is 15.22, which means that the average number of edges connected to a node is 15.22. Network density,that is the number of actual connected edges in a graph divided by the maximum number of theoretically possible edges, is 0.001 in this data. A value of 0.001 is a very small number, indicating that the graph is very sparse, meaning that only a very small percentage of all nodes are connected.

3.3 EDA

Before starting to build the model in earnest, we ran EDA to understand the target variables and the distribution of nodes and edges.



The 22,472 total nodes were not evenly distributed. There were 6,880 government-related pages, 6,495 company-related pages, 5,768 politics-related pages, and 3,327 TV show-related pages. In addition, the top three most distributed features in feature distribution EDA were feature3832, feature3818, and feature2032, with counts of 5,498, 3,836, and 3,208, respectively. The degree distribution roughly looks like a graph of the 1/x function. The highest degree was 1, and 2,660 nodes were connected to only 1 node.

4. DL Solutions

After building an ETL pipeline to prepare the source data into a form that can be used in the model, we applied machine learning (ML) solutions before using deep learning (DL) solutions for several reasons. First, ML models are generally simpler in structure and easier to interpret than DL models. It's easier to understand the basis of the model's judgment because you can see things like feature importance. Also, while DL requires high computing power due to its complex neural network structure, ML algorithms have a relatively low computational load. For these reasons, we used ML models as a baseline model to evaluate the performance of our main solution, the DL model.

4.1 ETL

We organized our ETL pipeline into four phases: data extraction, data transformation, data loading, and data partitioning. In the first data extraction step, we used two functions: the `get_features` function to extract the feature values corresponding to each node's ID from a dictionary, and the `get_edge_values` function to extract the edge values, which are the connections between nodes. The extracted node attributes and edge values are stored in a dataframe.

In the second data conversion step, we convert the previously extracted data into a form that can be input into the model. We vectorized the node attribute values into a one-hot encoding and converted them into PyTorch tensors. Edge indices are also converted to tensors, and node types, which are labels, are encoded as numbers.

In the third data loading step, we created a Data object from the PyTorch Geometric library to store the converted node attributes, edge indices, and label data.

In the final data partitioning step, we shuffled all the nodes randomly and split the training and test data with a certain ratio (0.8). We created train_mask and test_mask tensors to mask whether each node corresponds to training or test data.

4.2 ML models

We first converted the node attributes, labels, and training/test masks of the graph data from PyTorch tensors to NumPy arrays, which will allow us to implement the machine learning model in the scikit-learn library. We then split the data into training and test data using the training/test masks.

Next, we defined the machine learning classification models we wanted to use (**Logistic Regression**, **Random Forest**, and **Support Vector Machine**). For each classification model, we trained the model on standardized training data and made predictions on the test data. We then calculated the accuracy for each model and printed it out.

4.3 DL models

We implemented and trained GCN, GraphSAGE, and GAT models and measured the accuracy of each.

4.3.1 GCN

To implement the GCN model, we first defined a class, which inherits from PyTorch's torch.nn.Module, that defines two GCN layers. The first layer maps from the input feature count to 16 features, and the second layer maps from 16 features to the number of classes. ReLU is used as the activation function. The forward method returns the result after the data passes through each layer.

Next, we initialize the model and optimizer. The model is created as an instance of the GCN class, passing it the number of node features and the number of classes in the data. The optimizer is initialized using torch.optim.Adam. The hyperparameters of the optimizer are set to a learning rate of 0.01 and a weight reduction of 5e-4.

The learning loop trains the model by iterating over a specified number of epochs of 200. At each epoch, we initialize the gradient of the optimizer, pass the data to the model to get the output, calculate the loss, and perform backpropagation to update the model parameters.

4.3.2 GraphSAGE

Next, we defined the GraphSAGE model. The GraphSAGE class also consists of two GraphSAGE layers. Each layer maps the input features to a specific dimension, and then applies a ReLU activation function to introduce nonlinearity. We also applied dropout during training to prevent overfitting.

Next, we initialized the model and prepared it for training. The Adam optimizer is used, with the learning rate set to 0.01 and the weight reduction set to 5e-4, the same as the GCN model. These optimizers perform gradient-based optimization to update the model parameters.

The learning loop trains the model by iterating over 200 epochs, where each epoch computes the model's prediction, calculates the loss on the training data, performs backpropagation, and updates the model parameters.

4.3.3 GAT

Finally, we performed graph classification and node embedding extraction using GraphSage and Graph Attention Network (GAT) models.

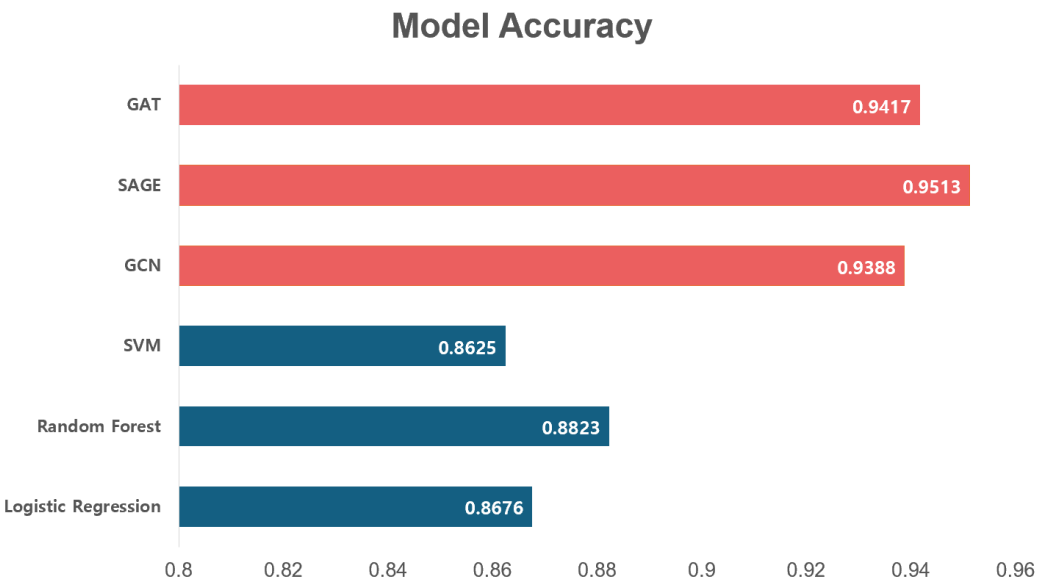
First, the `extract_node_embeddings` function extracts node embeddings using a given GraphSage model and data. It sets the model to evaluation mode and passes the graph features and edge indices to the first GraphSage layer of the model to compute the node embeddings.

It then defines a Graph Attention Network (GAT) model by defining a GAT class. A GAT uses multiple heads to compute the importance of a node's neighbors. We organized our class into two GAT layers, the first layer using multiple heads and the second layer using a single head.

We then set up an Adam optimizer and train the model for a given epoch. The hyperparameters of the optimizer and epoch are the same as in the previous model.

During training, we initialize the gradient of the optimizer, pass the node embeddings through the model to get the output, calculate the loss on the training data, and perform backpropagation to update the model parameters.

5. Results



From the resulting graph, we can see that the GraphSAGE model has the highest accuracy. With an accuracy of about 0.9513, it outperformed the other models. GAT had the second highest accuracy, with an accuracy of about 0.9417, performing close to the SAGE model. GCN had an accuracy of 0.9388. A detailed table of results, including other metrics besides the accuracy of the graph model, is attached in the appendix.
Meanwhile, traditional machine learning models had relatively low accuracy compared to graph neural networks.

6. Conclusion

In this project, we evaluated several graph neural network models, including GCN, GraphSAGE, and GAT, for classifying communities in the Facebook Large Page-Page Network dataset. The GraphSAGE model achieved the highest accuracy of around 0.9513, outperforming GCN and GAT as well as traditional machine learning models.
The strong results demonstrate the effectiveness of GNNs in leveraging graph-structured data like social networks. By capturing inter-node relationships, GNNs can better classify nodes/communities, enabling applications like targeted marketing, personalized recommendations, and enhanced customer segmentation. As social networks grow, GNN techniques hold significant potential for deriving valuable insights from

their interconnected structures.
In this project, the dataset only contained four labels, limiting its applicability to various business scenarios. Therefore, in future endeavors, the model will be expanded to detect pages that generate fake news or misinformation, as well as pages with harmful content. With the rise of generative AI, misinformation created by AI has the potential to impact individuals and even entire countries, thereby undermining the reliability of online platforms. Therefore, developing a harmful page detection model using GNN models could serve as a valuable business application for social networking platforms.

Appendix - Result Matrixes for GNN Models

GCN

	precision	recall	f1-score	support
0	0.9442	0.9393	0.9418	1334
1	0.9282	0.9589	0.9433	1362
2	0.9503	0.9572	0.9537	1098
3	0.9388	0.8771	0.9069	700
accuracy			0.9399	4494
macro avg	0.9404	0.9331	0.9364	4494
weighted avg	0.9400	0.9399	0.9397	4494

GraphSAGE

	precision	recall	f1-score	support
0	0.9644	0.9340	0.9490	1334
1	0.9347	0.9670	0.9506	1362
2	0.9584	0.9663	0.9624	1098
3	0.9417	0.9229	0.9322	700
accuracy			0.9502	4494
macro avg	0.9498	0.9475	0.9485	4494
weighted avg	0.9504	0.9502	0.9501	4494

GAT

	precision	recall	f1-score	support
0	0.9542	0.9363	0.9451	1334
1	0.9274	0.9567	0.9418	1362
2	0.9465	0.9499	0.9482	1098
3	0.9395	0.9100	0.9245	700
accuracy			0.9417	4494
macro avg	0.9419	0.9382	0.9399	4494
weighted avg	0.9419	0.9417	0.9417	4494

Code

https://colab.research.google.com/drive/13P5PHOSFvquCzCXusFLouad8Jayiz9ZY?usp=sharing

Citations

[1] Li, B., Nabavi, S. A multimodal graph neural network framework for cancer molecular subtype classification. *BMC Bioinformatics* 25, 27 (2024). https://doi.org/10.1186/s12859-023-05622-4

[2] A. K. Awasthi, A. K. Garov, M. Sharma and M. Sinha, "GNN Model Based On Node Classification Forecasting in Social Network," 2023 International Conference on Artificial Intelligence and Smart Communication (AISC), Greater Noida, India, 2023, pp. 1039-1043, doi: 10.1109/AISC56616.2023.10085118.

[3] https://snap.stanford.edu/data/facebook-large-page-page-network.html

[4] Rozemberczki, Benedek et al. "Multi-scale Attributed Node Embedding." ArXiv abs/1909.13021 (2019): n. pag.