Given Social circles (Facebook) Dataset Use Graph Neural Network To Build Friend Recommendation System

Project Report

Taolue CHEN	Mengyu LIANG	Nhat Mai NGUYEN	Yanjie REN	Jinji SHEN
CentraleSupélec	CentraleSupélec	CentraleSupélec	CentraleSupélec	CentraleSupélec
taolue.chen	mengyu.liang	nhatmai.nguyen	yanjie.ren	jinji.shen
@student-cs.fr	@student-cs.fr	@student-cs.fr	@student-cs.fr	@student-cs.fr

ABSTRACT

Friend recommendation is a key component for social media. We define the friend recommendation as a two-step problem that first predicts possible links among users using current social networks, and then selects k users with highest linking probability. To realize this, we applied 5 graph neural networks (GCN, GraphSAGE, GAT, GIN and Deep Linker) to link prediction on a subset of Stanford Social Circle Facebook dataset. We found that Deep Linker, a hybrid GAT model that encodes node information into vector representations and classifies links with logistic regression, achieves the best accuracy and AUC score while the rest models also provide satisfactory results in a short time. Our models could be generalized to larger datasets and efficiently give good results.

Keywords: friend recommendation, link prediction, graph neural network, attention mechanism

1. Introduction

Social media platforms such as Facebook, Twitter, and Instagram have revolutionized the way people interact with each other. They allow users to connect with friends and family members, as well as people with similar interests and backgrounds. One of the most critical components of social media platforms is their ability to recommend friends to users, which helps users to expand their social network and connect with people who share their interests and hobbies.

Friend recommendation systems are complex and building them requires advanced data technologies to identify meaningful relationships among users. With billions of users and interactions occurring daily, it can be challenging to create algorithms that accurately identify relevant relationships. Therefore, this project aims to build an advanced friend recommendation engine that uses cutting-edge machine learning techniques.

The primary problem we are trying to solve is building an effective friend recommendation system that can accurately identify meaningful relationships between users. This project will involve identifying the most relevant features for the recommendation engine, designing and training machine learning algorithms to predict connections among users, and implementing the recommendation system with a given real-world dataset.

The importance of this project lies in its potential to improve user engagement and retention on social media platforms. An effective friend recommendation engine will help users find relevant connections and expand their social network, leading to increased platform usage and a more engaged user base. Additionally, such recommendation systems have applications in social network analysis, marketing, and other data-driven fields, making this project a valuable contribution to the data science community.

Some potential applications of this project include: 1) Improving user engagement and retention on social media platforms; 2) Facilitating relevant connections between users with similar interests or backgrounds; 3) Identifying social network patterns and trends for marketing and research purposes; 4) Enhancing the user experience by providing more personalized content and recommendations.

In this report, we will firstly define the problem we are trying to solve, followed by a concise literature review of related works on friend recommendation. And then the detailed information about the methodology and evaluation methods we used will be introduced in the next two parts, with a brief conclusion at the end.

2. Problem definition

The problem addressed in this project is to develop an efficient and accurate friend recommendation engine using Graph Neural Networks (GNN) on the Stanford Social Circle Facebook dataset [4]. The goal of the system is to provide personalized and relevant friend recommendations to users of social media platforms, thus enhancing user engagement and retention.

To achieve this goal, the project will utilize advanced data technologies such as GNN models (GCN, SAGE, GAT, GIN, etc.) and other data science techniques to analyze user behavior and social connections within the dataset. The model will take into account the structure of the social network, including the nodes (users) and edges (connections between them), to make recommendations that are not only personalized but also relevant to the user's social context.

Formally, the problem can be defined as follows: Given a social network represented as a graph G = (V,E), where V denotes the set of nodes representing the users and E denotes the set of edges representing the connections between them, the objective is to predict the likelihood of a user u to form a meaningful connection with another user $v \in V$. The meaningful connection is defined as a connection that is likely to lead to engagement and interaction between the users. The system aims to optimize the accuracy of the recommendations, by minimizing the error function between the predicted and actual user connections.

The problem is computationally challenging due to the large scale of the dataset, with billions of users and interactions. Additionally, the problem involves processing and analyzing graph-structured data, which requires advanced machine learning techniques. Therefore, the project will address the problem by developing novel GNN-based algorithms, which can handle the complexity and heterogeneity of the social network data.

3. Related works

When building a friend recommendation system using Graph Neural Networks (GNN) on a social network dataset, it's important to understand the existing research in this area. Related works can be broadly categorized into two main groups: classic methods and GNN based methods.

Early papers on building recommendation systems using classic methods focused on using graph-based models to identify community structure in social networks and suggest friends within those communities. Traud et al. (2012) used data from Facebook to analyze the social structure of the online social network. They used classic network science methods to investigate the patterns of connection and found that Facebook users form clusters that are more tightly connected than would be expected in a random network [5]. The paper "Friendship prediction and homophily in social media" by E. Backstrom et al. (2011) uses graph-based approaches to predict future links in a social network based on homophily (the tendency of people to form ties with others who are similar to them). They find that homophily is a strong predictor of link formation, and that the effect is stronger in the IMDb network than in Facebook. They also develop a model that combines homophily with other factors, such as popularity and structural features of the network, to predict links in the future. [6]. However, these methods were still limited by the need for handcrafted features and the inability to incorporate node features beyond those explicitly defined.

More recently, GNN-based methods like GraphSAGE, GCN, GIN, and GAT have been proposed for friend recommendation systems. One paper that used GraphSAGE for friend recommendation was "Inductive Representation Learning on Large Graphs" by Hamilton et al. (2017). This paper proposed a combination of sampling and aggregation to generate node embeddings for large graphs. The authors evaluated GraphSAGE on a range of graph classification tasks and showed that it outperformed other existing methods. They also demonstrated that GraphSAGE can learn node embeddings for previously unseen nodes in the graph, making it a useful tool for tasks such as link prediction and recommendation systems [7]. Paper titled "PinSage: Graph Convolutional Neural Networks for Web-Scale Recommender Systems" by Ying et al. (2018) used PinSage, which combines efficient random walks and graph convolutions to generate embeddings of nodes that utilize both graph structure and node feature information [8].

Another paper by Zhang et al. (2019) proposed a GCN-based method called STAR-GCN for friend recommendation on Facebook, which employs a stack of GCN encoder-decoders combined with intermediate supervision to improve the final prediction performance. STAR-GCN learns low-dimensional user and item latent factors as the input to restrain the model space complexity. Moreover, STAR-GCN can produce node embeddings for new nodes by reconstructing masked input node embeddings, which essentially tackles the cold start problem [9]. A hybrid GAT model for link prediction, Deep Linker, is proposed by Gu et. al. (2019), which encodes each node in a link using a hybrid GAT and GraphSAGE architecture, and applies logistic regression that takes the encoding vectors as inputs and makes link predictions [10]. Additionally, a paper by Xiang Wang et al. (2018) proposed a model called Heterogeneous Graph Neural Networks (HGNN) that combines GIN with attention mechanisms to handle heterogeneous information in recommendation systems. They evaluate their model on several benchmark datasets and show that it outperforms several state-of-the-art recommendation algorithms [11]. Lastly, a paper titled "Session-based Recommendation with Graph Neural Networks" by Wang, H. et al. (2019) proposed a framework that leverages graph attention networks (GATs) to model

user-item interactions as a session-based graph. The GAT-based model captures the user-item interactions in the current session as well as the related items' information to predict the next item in the session. The authors evaluate their model on two real-world datasets and show that their proposed framework outperforms several state-of-the-art baselines [12].

These GNN-based methods have shown promising results in improving the performance of friend recommendation systems on Facebook, especially in addressing the cold start problem (a situation in which a recommendation system or machine learning model is unable to provide accurate predictions or recommendations for new users or items with no or limited data history) and incorporating node features beyond those explicitly defined.

4. Methodology

To address the problem of building a more accurate and relevant friend recommendation system using graph neural networks, we follow the following steps:

4.1 Data collection

The dataset used in this project, i.e. Social Circles (Facebook) dataset [4] consists of 'circles' or 'friends lists' obtained from survey participants through a Facebook app. It includes node features (profiles), circles, and ego networks. To preserve user privacy, the Facebook-internal ids for each user have been replaced with new values, and the feature vectors have been obscured. For example, a feature like "political=Democratic Party" in the original dataset would be replaced with "political=anonymized feature 1" in the anonymized data. Therefore, while it is possible to determine if two users share the same political affiliations, the actual meaning of individual political affiliations cannot be inferred from the anonymized data.

Dataset statistics	
Nodes	4039
Edges	88234
Nodes in largest WCC	4039 (1.000)
Edges in largest WCC	88234 (1.000)
Nodes in largest SCC	4039 (1.000)
Edges in largest SCC	88234 (1.000)
Average clustering coefficient	0.6055
Number of triangles	1612010
Fraction of closed triangles	0.2647
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7

Figure 1: Summary statistics of the dataset [4]

The term "Ego" refers to the individual who owns the network. The Ego user has the ability to create groups based on shared bonds and qualities with the users they follow, so he is connected with all other users in the network.

To maintain the simplicity of this project, we will solely focus on a single Ego network from the dataset, specifically the network belonging to **User 0**. The graph comprises 347 nodes and 5038 edges, and each node is represented by a 224-dimensional feature vector consisting of 0 and 1.

4.2 Data preprocessing

At first, we randomly shuffle the dataset and use a ratio of 0.3 to split the edges into training set and test set, meaning that 30% of the edges are used for testing, and 70% are used for training.

Next, in order to create a model that can better distinguish between real and fake edges more efficiently, we generate negative edges which represent the non-existent relationships or connections between nodes in the graph g. To do so, We tried to construct positive edges sets using the edges from the training set and negative edges by randomly selecting edges that do not exist in the graph. To be specific, a sparse matrix representation of the graph g is first created using the '*sp.coo_matrix'* function. The matrix is then negated and diagonal elements are set to zero. After this we use this new matrix to find the indices of all the nodes that are not connected by an edge in the original graph g, which correspond to the source and destination nodes of the negative edges.

Finally, the negative edges are randomly split into training and testing sets, both positive and negative graphs are then created for training and testing processes using the '*dgl.graph'* function, and we add self-loops to the training graph to ensure that each node is connected to itself.

4.3 Modeling

To build the Friend Recommendation System, we design 5 different architectures based on Graph Neuron Network (GNN), namely Graph Convolutional Networks (GCN), GraphSAGE, Graph Attention Networks (GAT), Graph Isomorphism Networks (GIN), and Deep Linker, which are popular and effective choices for learning node representations in graph data.

A GNN is a type of Neural Network that operates directly on the graph structure. The main concept behind GNN is node embeddings, which involves embedding a node with multiple layers of non-linear transformations based on the graph structure. GNN generates node embeddings based on local network neighborhoods and can be of arbitrary depth.

Each node has its embedding at each layer, where the input feature is the layer-0 embedding of a node. During each message-passing iteration, the embeddings are updated by message computation and aggregation. The message-passing operation is permutation equivariant, which enables every node to accumulate information in its local neighborhood.

Mathematically, GNNs learn node representations by performing graph convolutions on the adjacency matrix of the graph. The adjacency matrix represents the connections between nodes in the graph. The graph convolution operation updates node representations by taking into account the features of its neighboring nodes.

$$\mathbf{m}_u^{(l)} = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right)$$
$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}\right)$$

The algorithms used in the GNN models are variations of the basic graph convolution operation. The explanations of each model's algorithm are as follows.

GCN: a simple GNN architecture that learns node representations by aggregating information from the neighborhood of each node. It consists of multiple layers of graph convolutional operations, each of which updates node representations by taking into account the features of its neighboring nodes. The GCN model used in this implementation has two layers, both using the GraphConv module from the DGL library. This approach can be useful in predicting links between users who have interacted frequently in the past.

GraphSAGE: a popular GNN architecture that learns node representations by aggregating information from the neighbors of each node in a fixed-size neighborhood. It uses a "sampling" process to select a fixed number of neighbors for each node and aggregates their features to compute the node representation. The GraphSAGE model used in this implementation has two layers, both using the SAGEConv module from the DGL library. This approach can be useful in predicting links between users who have similar interests or who belong to the same community.

GAT: a variant of the Graph Attention Network architecture that learns node representations by attending to different parts of the neighborhood of each node, based on their importance. It uses a self-attention mechanism to weight the contributions of each neighboring node to the node representation, allowing it to effectively learn from different parts of the neighborhood. The GAT model used in this implementation has two layers and 8 heads, each layer using the GATConv module from the DGL library. This approach can be useful in predicting links between users who have a complex relationship in the social network.

GIN: a variant of the Graph Convolutional Network architecture that learns node representations by explicitly modeling the structural similarities between nodes. It uses a series of neural network modules to encode the local structure around each node, allowing it to capture more complex structural patterns in the graph. The GIN model used in this implementation has two layers, each using the GINConv module from the DGL library. This approach can be useful in predicting links between users who have a weak relationship in the social network.

Deep Linker: On the one hand, classic GAT models with numerous nodes, a small mini-batch size incorporates a significant number of nodes, decreasing the convergence speed and yielding inferior link predictions. On the other hand, classic GraphSAGE models change neighbors in each iteration which slows down the convergence and vibrates the losses. Deep Linker is composed of two GAT modules that each takes a node from a link as input and whose outputs (vector representations) are sent to a logistic regression for link classification. It intends to solve the memory limitation and convergence anomaly.

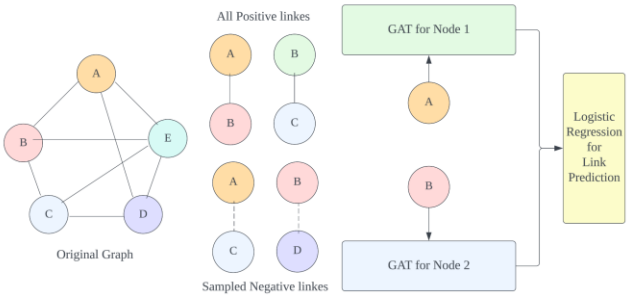

Figure 2: Architecture of the Deep Linker

Overall, each of these GNN architectures has its own strengths and weaknesses, and their performance depends on the specific characteristics of the social network data.

4.4 Hyperparameter Tuning

We also implemented hyperparameter tunings for GNN models to find the best combination of hyperparameters that produce the best performance for friend suggestions. In our

project, we tuned the hidden size and learning rate of the GNN models. The hidden size represents the number of hidden units in the GNN layer, and the learning rate represents the step size of each iteration while updating the parameters of the GNN models. To do so, we defined a 'tuning' function that takes the 'model_name' as an argument. Inside the function, it specifies a list of hidden sizes and learning rates to be tested for the given model name. The function then loops through all possible combinations of hidden sizes and learning rates, runs the pipeline function with those parameters, and saves the highest AUC and accuracy values achieved. After all iterations are completed, the function prints the best AUC and accuracy values along with their corresponding hyperparameters. Finally, the learned embeddings with the best AUC and accuracy scores are returned (See Figure 3).

Model	hidden_size	learning_rate
GCN	32	0.001
GraphSAGE	64	0.001
GAT (head=8)	64	0.001
GIN	32	0.001
Deep Linker (head=6)	32	0.0005

Figure 3: Results of Hyperparameter Tuning

4.5 Link Prediction

Once we have generated node embeddings using the classic GNN models, we need to translate this information into a measure of the likelihood that a friendship exists between two nodes. To achieve this, we can use a method called link prediction, which assigns a score to every possible edge in the graph based on the similarity of the embeddings of the nodes it connects. One way to compute this score is by taking the dot product of the embeddings of the two nodes (see figure 4). The DotPredictor module takes as input a graph g and the learned node representations h, and computes a new edge feature named 'score' by taking the dot product between the source node feature 'h' and destination node feature 'h'. The resulting 'score' feature represents the predicted likelihood of a link between the two nodes. However, this score is not a true probability since it can exceed the range of [0, 1], although it can be squished into this range using a sigmoid function.

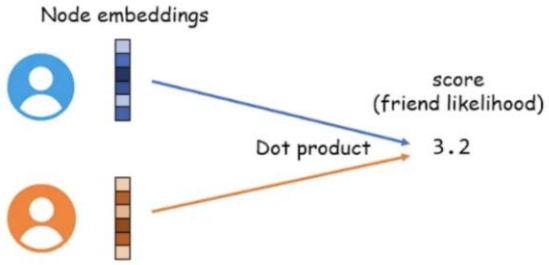

Figure 4: Friend Likelihood

On the other hand, as its structure suggests, the Deep Linker model is designed for link prediction tasks, thus we can use it directly to predict the friend likelihoods.

4.6 Rank potential friends

The next step is to generate a list of recommended friends for a particular user. This is achieved by calculating the similarity scores between the user and all other users who are not currently their friends. The list of users is then sorted by these scores, and the top K users are recommended to the user as potential friends. However, in practice, the K recommended users may not be the ones with the highest scores, but a random subset of 10 times K users with the highest scores, to provide a fresh set of recommendations to the user. The entire recommendation pipeline is depicted in the following figure.

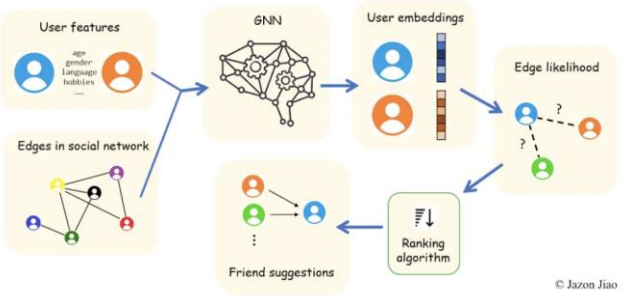

Figure 5: Recommendation Pipeline *(cf. Jazon Jiao)*

List of 5 suggested friends for User 0	
User id	Predicted Link Score
236	0.8474
169	0.8394
86	0.8177
241	0.8140
192	0.8099

Figure 6: One example of the output

4.7 Limitations

4.4.1 Data sparse problem

The problem of data sparsity is the lack of available information. First of all, most users in real social networks have no mutual friends, and only a few users have a certain number of mutual friends. Secondly, for users who are not highly active, they generate less content information, and such users seldom interact with other users. These factors lead to the problem of data sparseness in the process of friend recommendation. The hybrid friend recommendation method is one of the main solutions at present, and it is realized through the complementarity between different methods. In addition, some studies have overcome the data sparsity to a certain extent by expanding the depth of friend search and expanding the set of potential friends of users, and some studies have introduced ontology databases to expand rich text topics to solve the problem of sparse content data. In order to better understand the user's needs and extract the user's characteristic data, Zarrinkalam F et al. simulated the interest of inactive users based on the user's tendency to active topics on Twitter. However, the problem of data sparseness in friend recommendations is still a big challenge.

4.4.2 Computational costs

One limitation of GNN models is that they can be computationally expensive to train, especially for large graphs. To address this, Deep Linker is proposed, which uses two GAT modules to learn node representations for each node in a link and then sends these representations to a logistic regression to predict the likelihood of a new friendship between the two nodes. This approach reduces the computation required to train the model and improves its convergence speed.

4.4.3 Inability to use ranking-based metrics.

When evaluating a recommender system, there are two main categories of metrics: ranking-based and classification-based. We mentioned in the proposal that we will use HitRate@K as a ranking-based metric to the percentage of test cases where the recommended friends are included in the top K items that a user actually interacted with or showed interest in, as illustrated in the following example:

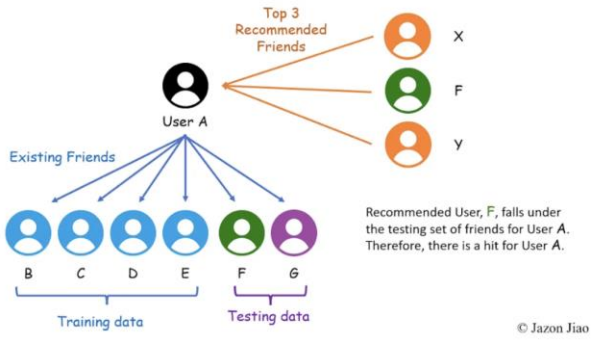

Figure 7: HitRate@K *(cf. Jazon Jiao)*

However, due to the random splitting of testing and training edges in our implementation method, it is not feasible to use ranking-based metrics like HitRate@K. This is because some users may not have any friends in the test set, which would result in them never receiving a hit in the HitRate@K metric while HitRate@K requires that a user has at least one friend in the test set to be able to calculate the metric accurately. Therefore, for this project, we will only use classification-based metrics to evaluate the performance.

4.4.4 Interpretability

Also, either classic GNN models or Deep Linker can yield high accuracies and AUC scores for link prediction, it is still unclear that why and how they could give good results. The mechanism is not yet explainable due to their black box processes.

5. Evaluation

To evaluate the performance of the recommendation system on the Facebook dataset, we conducted experiments on a

hold-out test set of user-user friend relationships. We split the data into a training set (70%) and a test set (30%). We trained the models on the training set and then evaluated their performance on the test set using various evaluation metrics.

We apply Accuracy and AUC (area under curve) to evaluate and compare the performances of these models for link prediction, whose tasks are binary classifications. We use accuracy to measure the proportion of correct predictions made by the model out of all predictions made, and use AUC to measure how well the model is able to distinguish between positive and negative classes.

Model	Accuracy	AUC
GCN	0.83	0.89
GraphSAGE	0.83	0.87
GAT	0.84	0.87
GIN	0.81	0.86
Deep Linker	0.85	0.92

Figure 8: Comparisons of Model Performance

Figure 8 illustrates that in classic GNN models, GCN is able to best distinguish negative and positive samples, while GAT achieves the highest accuracy score. Meanwhile, Deep Linker provides a state-of-the-art result for both accuracy and AUC score, suggesting its expertise in link prediction tasks.

6. Conclusion

In this project we studied various models for friend recommendation. In particular, we applied deep graph models ranging from GCN to Deep Linker to predict links with possible existence. We found that Deep Linker gives the relatively best performance considering accuracy and AUC score, although it takes longer time to run the epochs. Meanwhile, classic graph neural networks can yield accurate link predictions within a short period of time.

A noticeable point is that, with the property of fixed sampled neighbors, Deep Linker is able to mitigate the memory limitation, which is especially significant when there is a huge number n of nodes and the complexity of link prediction is $O(n^2)$. Thus, in the further research in which the size of the graph is considerably large (for instance, 5,000 nodes with 80,000 edges), predicting potential links with Deep Linker model could be more efficient since it can reduce GPU memory usage and accelerate the training process.

Despite good performance on link prediction, from a business perspective, we want to go further with our models by studying the mechanisms and working on their interpretability. We believe that such an interpretable and well-performing model could be a credible choice for social media and platforms.

REFERENCES

[1] Schröder, G., Thiele, M., & Lehner, W. (2011, October). Setting goals and choosing metrics for recommender system evaluations. In UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA (Vol. 23, p. 53).

[2] Salamat, A., Luo, X., & Jafari, A. (2021). HeteroGraphRec: A heterogeneous graph-based neural networks for social recommendations. Knowledge-Based Systems, 217, 106817.

[3] Sharma, K., Lee, Y. C., Nambi, S., Salian, A., Shah, S., Kim, S. W., & Kumar, S. (2022). A Survey of Graph Neural Networks for Social Recommender Systems. arXiv preprint arXiv:2212.04481.

[4] Leskovec, J., & Mcauley, J. (2012). Learning to discover social circles in ego networks. Advances in neural information processing systems, 25. Available at: http://snap.stanford.edu/data/egonets-Facebook.html

[5] Traud, A. L., Mucha, P. J., & Porter, M. A. (2012). Social structure of facebook networks. Physica A: Statistical Mechanics and its Applications, 391(16), 4165-4180.

[6] Backstrom, E., Boldi, P., Rosa, M., Ugander, J., & Vigna, S. (2011). Friendship prediction and homophily in social media. ACM Transactions on the Web (TWEB), 5(1), 2.

[7] Hamilton, W., Ying, R., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. In Advances in Neural Information Processing Systems (NIPS) (pp. 1024-1034).

[8] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). PinSage: Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In Proceedings of the 2018 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 135-144).

[9] Zhang, J., Shi, X., Zhao, S., & King, I. (2019). Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems. arXiv preprint arXiv:1905.13129.

[10] Gu, W., Gao, F., Lou, X., & Zhang, J. (2019). Link prediction via graph attention network. arXiv preprint arXiv:1910.04807.

[11] Zhang, H., Cui, Z., Neumann, M., & Chen, Y. (2019). GIN: Graph convolutional networks for web-scale recommender systems. In Proceedings of the 13th ACM Conference on Recommender Systems (pp. 2-10). arXiv:2009.00799v1

[12] Wang, H., Zhang, F., Liu, X., & Sun, M. (2019). Session-based Recommendation with Graph Neural Networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 346-353).