GATRec: A Graph Attention Network Approach to Academic Paper Recommendation

Zhennan Shen* Bo Huang*

Shanghai Jiao Tong University, Shanghai, China
ieee-szn@sjtu.edu.cn, huang20003bo@sjtu.edu.cn

Abstract. Personalized recommendations constitute an important feature of modern online services, including major e-commerce platforms, social media, and streaming services. The rapid development of Graph Neural Networks (GNNs) has demonstrated their strong expressive ability to capture high-order connectivity in user-item interaction data. The academic recommendation scenario can effectively leverage heterogeneous graphs for modeling, thus we adopted a graph-based recommendation approach using heterogeneous graphs. We incorporate the currently popular Attention mechanism to better learn the features in User-Item interactions. Our experimental results show significant improvements in recommendation accuracy and relevance, outperforming several state-of-the-art baseline, by achieving the accuracy of 0.94324. Our code is available at `https://github.com/5456es/GATRec`, more information about the competition can be seen at `https://www.kaggle.com/competitions/cs3319-01-project-2-recommendation-2024-spring`.

Keywords: Graph Attention Network · Academic Paper Recommendation · Link Prediction

1 Introduction

Graph Neural Networks (GNNs) have firmly established themselves as state-of-the-art approaches in numerous graph-based tasks, including node classification, graph generation, and link prediction. These tasks are gaining popularity due to the inherent graph structure of many societal frameworks and their increasing digitalization.

One prominent application of graph-based tasks is in recommender systems, which have become particularly popular with the advent of the big data era. Modern e-commerce and social media platforms extensively use recommender systems to provide personalized product suggestions and notifications for potentially interesting content. These systems not only mitigate the problem of information overload but also enable users to discover items of potential interest more efficiently, enhancing the overall user experience.

In this paper, we present our implementation of an academic paper recommender system. More specifically, by using a graph that encompasses citation relationships between papers, citations by authors, and co-author relationships

among authors, we employ Graph Neural Networks (GNNs) to predict papers that may be of potential interest to a specified author.

2 Related Work

Graph Neural Networks (GNNs) have emerged as a cornerstone in recommendation systems due to their proficiency in modeling intricate user-item interactions. The general framework for leveraging GNNs in recommendation systems typically encompasses three primary stages: initially constructing a heterogeneous graph from user-item data, subsequently employing graph aggregation techniques to derive encoded features for nodes from their respective neighbors, and ultimately predicting potential user-item interactions based on the learned node features. Below, we delve into four specific methods that adhere to this framework.

2.1 GC-MC

GC-MC employs a heterogeneous-graph-based auto-encoder framework specifically designed for matrix completion. This auto-encoder is trained using GNN methodologies, facilitating the generation of latent features for both user and item nodes through a process of message passing on the heterogeneous graph. A bilinear decoder is then utilized to reconstruct recommendation links based on these latent features, effectively capturing the underlying patterns in user-item interactions.

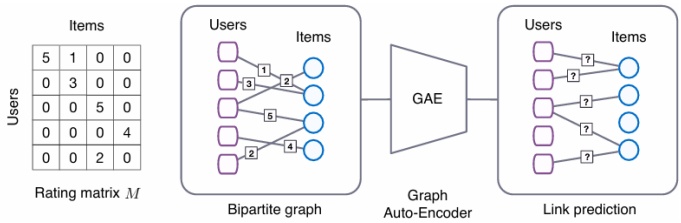


Fig. 1: Overview of GC-MC structure

2.2 NGCF

Xiang Wang et al. introduced NGCF, a method that integrates user-item interactions, particularly the bipartite graph structure, into the embedding process for learning vector representations of users and items within recommendation contexts. NGCF draws inspiration from the Graph Convolutional Network (GCN), adhering to a similar propagation rule that involves feature transformation, neighborhood aggregation, and nonlinear activation. By propagating

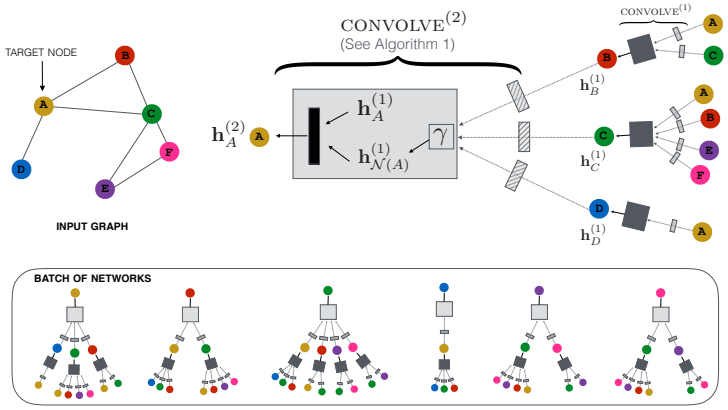


Fig. 2: Overview of PinSage structure using depth-2 convolutions (best viewed in color).

embeddings across the user-item graph, NGCF adeptly captures high-order connectivity within the graph, explicitly incorporating collaborative signals into the embedding process and thereby enhancing recommendation accuracy.

2.3 LightGCN

In response to the complex and cumbersome design of NGCF, Xiangnan He et al. proposed LightGCN, a simplified yet effective modification that retains the fundamental architecture of GCN. LightGCN emphasizes the most critical component of GCN—neighborhood aggregation—while eliminating superfluous features. Specifically, LightGCN assigns each user and item an ID embedding and propagates these embeddings on the user-item interaction graph to refine them. The embeddings learned across different propagation layers are subsequently combined using a weighted sum to derive the final embedding for prediction. This streamlined approach significantly reduces computational overhead while maintaining, or even improving, the quality of recommendations.

2.4 PinSage

Recognizing the limitations of preceding GNN-based recommendation methods concerning practicality and scalability for web-scale recommendation tasks, Rex Ying et al. introduced PinSage. PinSage innovatively combines efficient random walks with graph convolutions to produce embeddings for items within a large-scale recommender system. One of the most novel aspects of PinSage lies in its use of highly efficient random walks to structure the convolutions, alongside a unique training strategy that employs increasingly challenging training examples to enhance the model's robustness and convergence.

3 Method

3.1 Overview

In our approach, there are mainly three process, preprocess of node feature, GNN extracting the hidden information of the potential relationship between nodes and the prediction of the potential link based on the information generated by previous two steps.

3.2 Preprocess of node feature

The effectiveness of embeddings has been consistently demonstrated across various fields, most notably with *word2vec* in Natural Language Processing (NLP). In the context of graphs, the analogous concept to word embeddings is node embeddings. Node embeddings aim to represent nodes in a continuous vector space, capturing the graph's structural and relational properties.

In this work, we utilize **MetaPath2Vec**, a method specifically designed for heterogeneous information networks. MetaPath2Vec leverages metapaths—sequences of node and edge types—to capture rich semantic relationships between different types of nodes. By performing random walks guided by these metapaths, MetaPath2Vec generates a corpus from which node embeddings are learned using a skip-gram model.

Algorithm 1 MetaPath2Vec Pseudocode

Require: Heterogeneous graph G, Metapath schema M, Embedding dimension d
Ensure: Node embeddings matrix E
Initialize E with random values
for each node v in G **do**
  $walks \leftarrow \text{RandomWalk}(G, v, M)$
  for each walk in $walks$ **do**
    $pairs \leftarrow \text{GenerateContextPairs}(walk)$
    Update E using Skip-gram on $pairs$
  end for
end for
return E

Algorithm 2 RandomWalk

Require: Graph G, Node v, Metapath M
Ensure: A sequence of nodes
1: $walk \leftarrow [v]$
2: **for** $i \leftarrow 1$ to l **do**
3: $neighbors \leftarrow \text{FilterByMetapath}(G.neighbors(walk[-1]), M[i \mod |M|])$
4: $walk.append(\text{RandomChoice}(neighbors))$
5: **end for**
6: **return** $walk$

3.3 GNN

GNN is the backbone of our approach. More specifically, we utilize the GAT as our backbone to extract potential information from the graph. Generally saying, GNN utilize the neighbour nodes' information to infer the current node. Here the GAT could be mainly considered as the GCN, the graph convolution network, with the sampler and attention mechanism.

In GCN, each node's feature representation is updated by taking the average of its neighbors' feature representations. The formula is given by:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{d_i d_j}} W^{(l)} h_j^{(l)}\right)$$

where $\mathcal{N}(i)$ denotes the set of neighbors of node i, d_i and d_j are the degrees of nodes i and j respectively, $W^{(l)}$ is the weight matrix at layer l, and σ is the activation function.

However, considering all the neighbors is computationally feasible for relatively small graphs, but the cost becomes prohibitive for larger graphs. To address this, we can revise the formula using a sampling mechanism:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{S}(i)} \frac{1}{\sqrt{d_i d_j}} W^{(l)} h_j^{(l)}\right)$$

where $\mathcal{S}(i)$ denotes a sampled subset of $\mathcal{N}(i)$.

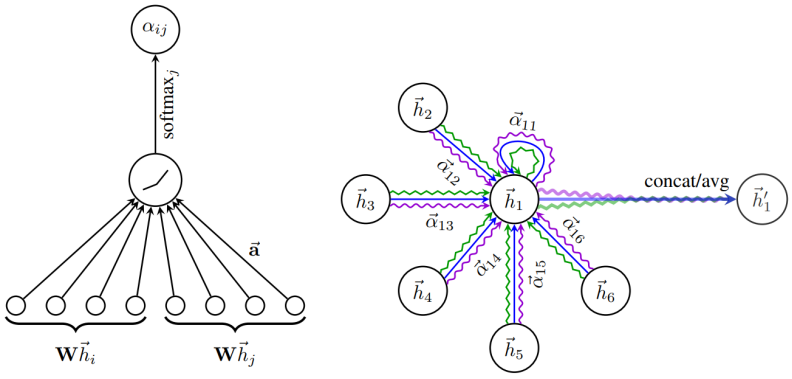
Further, we can improve the model by incorporating an attention mechanism to weigh the importance of each neighbor. This leads to the Graph Attention Network (GAT) formulation, where the update rule becomes:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{S}(i)} \alpha_{ij} W^{(l)} h_j^{(l)}\right)$$

Here, α_{ij} is the attention coefficient computed as:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(a^T [W^{(l)} h_i^{(l)} \| W^{(l)} h_j^{(l)}]))}{\sum_{k \in \mathcal{S}(i)} \exp(\text{LeakyReLU}(a^T [W^{(l)} h_i^{(l)} \| W^{(l)} h_k^{(l)}]))}$$

where a is a learnable weight vector, and $\|$ denotes concatenation.



3.4 Prediction of the potential link

- **Calculating Similarity:** For each pair of nodes u and v in the graph, we calculate the similarity score as the dot product of their embeddings: $s_{uv} = h_u \cdot h_v$.
- **Binary Prediction:** We convert the similarity scores into binary predictions. By trying different thresholds, we determine the optimal threshold that maximizes the F1 score, balancing precision and recall for the link prediction task.

Algorithm 3 Optimize F1 Score With Float Threshold

Require: True labels, Predictions
Ensure: Best F1 score, Best threshold
Initialize $best\_f1\_score = 0$, $best\_threshold = 0$
for each adjustment in range(Bottom, Top) **do**
  current_threshold ← compute_threshold(adjustment)
  adjusted_predictions ← adjust_predictions(predictions, current_threshold)
  f1_score ← evaluate_metrics(true_labels, adjusted_predictions)
  if $f1\_score > best\_f1\_score$ **then**
    $best\_f1\_score \leftarrow f1\_score$
    $best\_threshold \leftarrow current\_threshold$
  end if
end for

This process allows us to effectively predict potential links using the node embeddings extracted by GAT, revealing previously unknown relationships in the graph.

Algorithm 4 Link Prediction

Require: Node embeddings H, Similarity threshold T
Ensure: Predicted binary link matrix P
1: **for** each pair of nodes u and v **do**
2: Calculate similarity score $s_{uv} = H_u \cdot H_v$
3: $P_{uv} = \begin{cases} 1, & \text{if } s_{uv} \geq T, \\ \text{otherwise} \end{cases}$
4: **end for**
5: **return** P

Our training objective is to maximize the accuracy of link prediction. The fundamental form of the loss function is given by:

$$\mathcal{L} = -\sum_{e_i \sim \text{pos}} p(e_i) + \sum_{e_j \sim \text{neg}} p(e_j)$$

where $p(e)$ denotes the cosine similarity of the sampled edge. The positive sampler selects edges from the existing connections, while the negative sampler draws from the unlabelled edges.

This basic form can be adapted by altering the specific loss formulation (e.g., hinge loss) and modifying the sampling strategies for both positive and negative samples.

3.5 Other Mentionable Details

Unbiased Sampler In the context of using Graph Neural Networks (GNNs) for link prediction, a fundamental challenge arises in the treatment of unlabelled data. Specifically, in our dataset (bipartite_train_ann.txt), labelled data indicate existing links between authors and papers, whereas any author-paper pair not present in this dataset represents unlabelled data. While most of these unlabelled author-paper edges are likely true negatives, some may actually be positive samples that should be recommended.

Treating all unlabelled data as negative samples introduces bias into the loss function. To address this issue, we developed an improved sampler, termed the Unbiased Sampler. This sampler leverages a pretrained model to guide the negative sampling process, thereby aiming to more accurately reflect the true nature of the unlabelled data. Our experiments demonstrated that addressing the bias in the loss function is crucial, although it may not be the primary bottleneck in improving our model.

Residual Block Residual connections work by providing direct pathways for information and gradients to flow through the network, making it easier to train deep neural networks, improving gradient flow, and preventing issues such as vanishing gradients and degradation of performance. This allows neural networks

to be both deeper and more effective, leading to significant advancements in various fields of artificial intelligence and machine learning.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{1}$$

Where: $\mathbf{x}$ is the input. $\mathcal{F}(\mathbf{x}, \{W_i\})$ is the residual function learned by the block, with parameters $\{W_i\}$. $\mathbf{y}$ is the output.

In the context of a graph neural network (GNN) which is our task, residual connections help to preserve node features and facilitate the flow of information across layers. This is particularly beneficial for capturing long-range dependencies and improving the overall representation learning on graphs.

Voting Method The voting method is a technique used in ensemble learning where multiple models are employed to make predictions on the same task. The final prediction is determined by aggregating the individual predictions of these models, typically through majority voting. This method enhances the robustness and accuracy of the predictions by leveraging the diversity of the models.

4 Experiment

4.1 Task Overview

The task involves formulating an academic reading recommendation system to suggest relevant papers to authors based on their previous research. The dataset includes authors and papers from top GeoScience journals, encompassing citation information. A heterogeneous network is constructed with two types of nodes: authors and papers. In this network:

- **Author-Paper Edges:** Indicate that authors have read the papers (representing citations by the authors in their own papers).
- **Author-Author Edges:** Represent co-authorship relationships.
- **Paper-Paper Edges:** Are directed and represent citation relationships between papers.

The objective is to predict whether a paper should be recommended to an author, treating this as a link prediction problem. A recommended paper is marked as 1, while a non-recommended paper is marked as 0. This approach leverages the inherent relationships and connections within the academic network to generate relevant paper recommendations for authors.

4.2 Sampler

The DGL library provides two negative samplers for heterogeneous graphs. Based on the property of semi-supervised learning, we put forward a new score sampler to balance the attention between labeled and unlabeled data.

To effectively handle the link prediction task, we utilized two negative sampling strategies:

PerSource Uniform Negative Sampler This sampler randomly selects negative destination nodes for each source node based on a uniform distribution. For each edge (u, v), it generates k negative edges (u, v'), where v' is uniformly chosen from all nodes of the same type as v.

Global Uniform Negative Sampler This sampler randomly selects negative source-destination pairs uniformly. For each edge (u, v), it generates up to k negative edges (u', v'), with u' and v' chosen uniformly from all nodes of their respective types. It can exclude self-loops and sample with or without replacement.

The loss function based on the Global Uniform Negative Sampler(classical loss function) can be formed as

$$\mathcal{L} = -\sum_{e_i \in E(G)} p(e_i) + \sum_{\bar{e}_j \sim \bar{E}(G)} p(\bar{e_j})$$

where $p(e_i)$ denotes the probability of link e_i exists, E is the edge set of the graph, $\bar{E} := \{(u, v)|u \in V(G), v \in V(G), (u, , v) \notin E(G)\}$ $\bar{E}$ is the edge set of the complement graph of G.

The classical loss function introduces bias by considering all edges in $\bar{E}$ as negative samples. This approach inherently lowers the probability of link prediction for these edges. However, some edges within the unlabelled data should actually be considered positive samples. The bias of the classical loss function lies in $\sum_{e_j \sim (\bar{E} - E_{neg})}$, where:

$$E_{pos} := \{(u, v)|\text{there should be a link between u and v}\}$$

$$E_{neg} := \{(u, v)|u, v \in V(G), (u, v) \notin E(G)\}$$

E_{pos} and E_{neg} represents the ground truth of this task.

To measure the influence of the affect of the loss function bias, we statistically analyzed the possible positive edges in the complementary graph, denoted as $\frac{|\bar{E} - E_{neg}|}{|\bar{E}|}$

We assumed that the ratio of labelled training edges to all edges that should exist is consistent between the test set and the entire graph. This assumption is represented as:

$$\frac{|E_{test} \cap E_{train}|}{|E_{test} \cap E_{pos}|} = \frac{|E_{train}|}{|E_{pos}|}$$

Using a pretrained model that achieved a 94% F1 score on the test dataset, we estimated $|E_{test} \cap E_{pos}| \approx 10224159$. Consequently, we derived the ratio:

$$\frac{|\bar{E} - E_{neg}|}{|\bar{E}|} = \frac{3225588682421}{661179937682421} = 0.4\%$$

This indicates that the bias introduced by the classical loss function is minimal and thus tolerable within the context of our experiments.

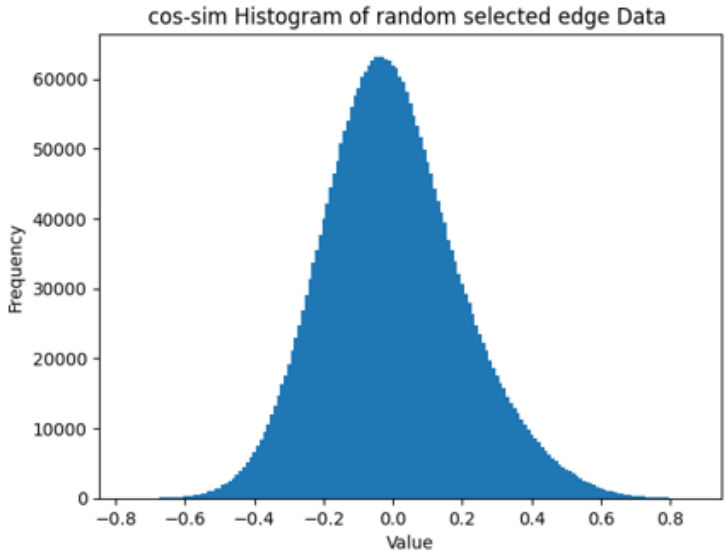


Fig. 3: Histogram of Cosine Similarities for Randomly Selected Edges. The histogram shows the distribution of cosine similarities for edges randomly selected between nodes in the graph. Most edges have cosine similarities concentrated between -0.2 and 0.2. Edges with cosine similarities higher than 0.55 are rare, comprising approximately 0.0068186 of all edges.

Score Sampler Based on our bias analysis, we proposed a novel sampling approach to better address the potential positive samples in the unlabelled data. This sampler modifies the negative sampling process to appropriately suppress the probability of sampling edges that are highly likely to be positive.

Specifically, we calculate the ratio of positive edges among all possible links between any two nodes:

$$\frac{E_{pos}}{|\{(u,v)|u,v \in V\}|} \approx 0.006110$$

Using a well-pretrained model, we evaluated the cosine similarity distribution of randomly selected edges. The data indicated that the probability of a random edge having a cosine similarity higher than 0.55 is approximately 0.0068186.

From this, we concluded that any random edge should be considered a positive sample if its cosine similarity, as determined by the pretrained model, is higher than 0.55. Therefore, we developed a new sampler, termed the Unbiased Sampler, based on the Global Uniform Sampler. This new sampler reduces the likelihood of sampling edges with a cosine similarity higher than 0.55.

Experiments over the global uniform sampler and the unbiased sampler are listed as below.

Table 1: Results over **Sampler usage** with default configure[1]and not using residual block

	Model	Precision	Recall	F1-score
Global Uniform Sampler	**4-GAT**	0.9434	0.9437	0.94354
Unbiased Sampler	**4-GAT**	0.9435	0.9443	**0.94391**

4.3 Residual Block

Residual connections in neural networks solve the issues of information loss and gradient problems in deep networks. By adding a shortcut between each layer's input and output, they bypass nonlinear transformations.

This helps retain important information and alleviates vanishing or exploding gradients, improving network performance.

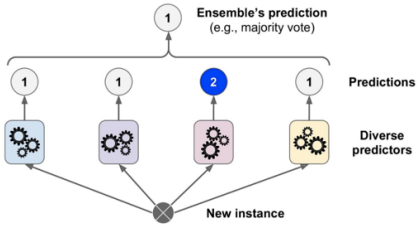
Table 2: Results over **Residual Block usage** with default configure and using Global Uniform sampler

	Model	Precision	Recall	F1-Score
w/o Residual	**4-GAT**	0.9424	0.9406	0.94149
w/ Residual	**4-GAT**	0.9474	0.9359	**0.94160**

4.4 Voting Method

The vote method in ensemble prediction involves combining the outputs of multiple models to make a final prediction based on a majority vote. Each model in the ensemble makes its own prediction, and the prediction that receives the most votes is chosen as the final output.

This approach leverages the strengths of multiple models to improve overall prediction accuracy and robustness.



[1] Default configure: epoch: 200, batch size: 10000, learning rate: 0.0001-0.00005 (using cosine decay, period 40), k=4, Dim=[512, 256, 256, 64], heads=[16, 8, 8, 4], weight decay=4e-5.

4.5 Batch Size & Learning rate & Model architecture

Modify the common hyperparameters:

	Batch Size	Precision	Recall	F1-Score
3-GCN	**10000**	0.9336	0.9238	0.92867
	15000	0.9334	0.9274	0.93039
	20000	0.9341	0.9236	0.92882
	Batch Size	**Precision**	**Recall**	**F1-Score**
4-GAT	**10000**	0.9521	0.9346	0.94327
	15000	0.9525	0.9335	0.94291
	20000	0.9531	0.933	**0.94294**

Table 3: **Learning rate:0.001 ∼ 0.0001 T=40**

	Batch Size	Precision	Recall	F1-Score
3-GCN	**10000**	0.9381	0.9234	0.93069
	15000	0.9366	0.9287	0.93263
	20000	0.9376	0.9257	0.93161
	Batch Size	**Precision**	**Recall**	**F1-Score**
4-GAT	**10000**	0.955	0.9326	0.94367
	15000	0.9543	0.9333	**0.94368**
	20000	0.9526	0.9343	0.94336

Table 4: **Learning rate:0.0005 ∼ 0.00005 T=40**

	Batch Size	Precision	Recall	F1-Score
3-GCN	**10000**	0.9373	0.9306	0.93394
	15000	0.9391	0.9287	0.93387
	20000	0.9388	0.9292	0.93398
	Batch Size	**Precision**	**Recall**	**F1-Score**
4-GAT	**10000**	0.9555	0.9322	0.94371
	15000	0.9543	0.9336	0.94384
	20000	0.9522	0.936	**0.94403**

Table 5: **Learning rate:0.0001 ∼ 0.00001 T=40**

5 Conclusion and Discussions

5.1 Final Performance

We applied a Graph Attention Network (GAT) for academic paper recommendations, generating author and paper embeddings using MetaPath2Vec. Utilizing Graph Convolutional Networks (GCNs) as our backbone, we incorporated attention mechanisms and integrated residual connections along with an unbiased sampler. To further enhance the overall accuracy and robustness, we employed a voting method. Our GATRec model finally achieved a remarkable F1 score of **0.94903**.

5.2 Reflection

Lack of Data Analysis and Understanding Challenge: Our experiment encountered challenges due to insufficient analysis and consideration of the graph data. Specifically, the unique characteristics of our bipartite author-paper graph and the distribution of positive and negative samples were not fully understood and leveraged.

Consequences: This deficiency resulted in an approach that often resembled a black-box optimization process. Without a deep understanding of the data, we were unable to effectively interpret and utilize it to guide the training process. This limitation may have hindered our ability to fine-tune our Graph Neural Network (GNN) architecture and sampling strategies optimally.

Lack of Model Performance Analysis Challenge: We did not perform an in-depth analysis of the performance of our GNN-based model on this specific task, particularly in terms of how different hyperparameters and architectural choices influenced our results.

Consequences: This oversight made it difficult to evaluate and fine-tune the neural network, creating significant challenges in optimizing the model's performance. Consequently, our ability to improve metrics like precision, recall, and F1-score was limited.

5.3 Future Work

To further enhance the capabilities of graph neural networks (GNNs) in academic recommendation systems, several avenues for improvement can be pursued:

Enhanced Data Analysis Visualizing node pair neighborhoods within the train/test data will be crucial. Specifically, comparing the neighborhoods of node pairs with and without established links aims to uncover local graph topologies. This analysis is essential for understanding task-specific performance boundaries, thereby guiding more effective training strategies.

Comprehensive Model Evaluation Continuous evaluation of model characteristics, including rigorous assessment of overfitting tendencies and performance bottlenecks, is imperative. This approach ensures early identification and mitigation of factors limiting model performance, thereby enhancing overall prediction accuracy and reliability.

Structured Hyperparameter Tuning Adopting systematic methods such as grid search or Bayesian optimization for hyperparameter tuning will streamline the configuration process. By systematically exploring the hyperparameter space, this approach aims to achieve more optimal model configurations, leading to improved prediction capabilities and efficiency.

Advancements in Model Architecture Exploring more sophisticated GNN models, such as Hierarchical Attention Networks (HAN) and Graph Attention Networks (GAT) with deeper architectures and diverse attention mechanisms, presents an opportunity for significant enhancement. This exploration aims to leverage advanced model structures to elevate both performance metrics and robustness in handling complex academic recommendation scenarios. These strategies collectively aim to advance the field of GNN-based academic recommendation systems, enhancing their effectiveness and applicability in real-world settings.

References

1. veličković, , P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.
2. Ziniu Hu and Yuxiao Dong and Kuansan Wang and Yizhou Sun. (2020). Heterogeneous Graph Transformer. arXiv preprint arXiv:2003.01332
3. Xiao Wang and Houye Ji and Chuan Shi and Bai Wang and Peng Cui and P. Yu and Yanfang Ye.(2021). Heterogeneous Graph Attention Network. arXiv preprint arXiv:1903.07293
4. William L. Hamilton and Rex Ying and Jure Leskovec.(2018). Inductive Representation Learning on Large Graphs. arXiv preprint arXiv:1706.02216
5. Rianne van den Berg and Thomas N. Kipf and Max Welling.(2017). Graph Convolutional Matrix Completion. arXiv preprint arXiv:1706.02263
6. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020, July). Lightgcn: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (pp. 639-648).
7. Ying, Rex and He, Ruining and Chen, Kaifeng and Eksombatchai, Pong and Hamilton, William L. and Leskovec, Jure.(2018, July). Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 974-983).