



香 港 大 學

THE UNIVERSITY OF HONG KONG

COMP4805 Project

Denoised Graph Contrastive Learning for Recommendation

Supervisor: Dr. Huang, Chao
Student: Xu Chen
University Number: 3035772597
Date: December 14, 2023

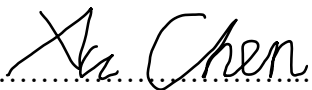
Department of Computer Science

ABSTRACT

As deep learning technology develops, Graph Neural Networks(GNN) have recently emerged as an effective approach in recommending systems. Besides, contrastive learning has manifested great effectiveness in learning general features from unlabeled data. In most recommending systems, user-item interaction data are of high sparsity, and graph contrastive learning has proved to be highly effective in dealing with the data sparsity issue. However, noises in user-item interactions may greatly affect the robustness and effectiveness of recommending systems in real-world applications. Meanwhile, several existing models did not extract contrastive information from cross-layer embeddings, which may waste valuable data among GNN layers. Therefore, this project aims to propose possible solutions to noise issues and data sparsity by applying a graph denoising module on input user-item data and adding cross-layer contrastive loss to the recommending model. Through extensive experiments on different datasets, the results show improvements in the robustness and accuracy of recommending systems, indicating the model proposed by this project may be more robust against sparse and noisy user-item interactions. The project can be found in the link: <https://comp4805projectxuchen.wordpress.com/denoised-graph-contrastive-learning-for-recommendation/>

DECLARATION

I declare that this report represents my own work, except where due acknowledgement is made, and that it has not been previously included in any report.

Signed 

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my project supervisor Dr. Huang Chao, who inspired my interest and guided me in learning Graph Neural Networks and Recommendation Systems.

TABLE OF CONTENTS

ABSTRACT.....	2
DECLARATION	3
ACKNOWLEDGEMENTS.....	4
CHAPTER 1 INTRODUCTION	6
1.1. Project Background.....	6
1.2. Motivation.....	6
1.3. Report Structure.....	7
CHAPTER 2 PROBLEM ANALYSIS.....	8
2.1. Basic Variables Definition.....	8
2.2. Related Concepts	8
2.3. Baseline Model: LightGCL	9
CHAPTER 3 METHODOLOGY	11
3.1. Denoising Module.....	11
3.2. Cross-Layer Contrastive Loss.....	12
3.3. Add Noises to Embeddings.....	13
CHAPTER 4 EXPERIMENTAL ANALYSIS.....	15
4.1. Experiment Settings.....	15
4.2. Hyperparameter Analysis	15
4.3. Comparative Analysis.....	17
CHAPTER 5 CONCLUSION	19
REFERENCES	20

CHAPTER 1

INTRODUCTION

1.1. Project Background

Nowadays, recommending systems have been widely applied to a large variety of web applications and online platforms, including news websites (Yang et al., 2023), video-sharing platforms(Wei et al.,) and online shopping malls(Wang et al., 2020), providing users with personalized and suitable recommendations. Meanwhile, deep learning technology is developing fast, which enables recommending systems to learn abundant and complicated user/item features, especially through graph neural networks (GNN). Given that user-item interactions can be easily formed using a graph, plenty of recent research about recommending systems adopt GNN-based Collaborative Filtering models as baselines, which have greatly improved recommendation performance (Tian et al., 2022). Moreover, the contrastive learning method also become widely applied in many recommendation models due to its ability to solve data sparsity issues. Most contrastive learning recommending models adopt a two-step approach: First, they perform user-item graph augmentation, usually by node or edge dropout; Second, from different augmented graphs, these models maximize the consistency of user/item embeddings learned through a joint learning framework.

1.2. Motivation

Despite the GNN-based contrastive learning being effective in dealing with sparse user-item data, many existing methods seem to haven't tackled the noise issue in the original interaction records. In the real world, user behaviors may be of high randomness, or due to erroneous events, which may not imply users' real preferences (Tian et al., 2022). More importantly, well-crafted fake interactions added by malicious users may even make the model biased (Zhang et al., 2021). Training a GNN-based recommending model with noisy data may affect the robustness significantly, because

GNN adopts a message-passing approach by aggregating each node’s neighborhood information iteratively (Tian et al., 2022), resulting in high vulnerability of GNN models against noisy data.

Meanwhile, many contrastive recommending models only calculate contrastive loss across different augmented graphs, but neglect the contrastive information for embeddings among layers. One method proposed by Yu et al.(2023) adopted a cross-layer contrastive learning approach, which was both efficient and effective. Inspired by this, this project plans to combine the contrastive learning approaches both for cross-layer and different augmented graphs.

Besides the above two methods, this project also implements an embedding augmentation, directly adding noise vectors to embeddings instead of changing graph structure. Initially, this idea was inspired by adversarial examples constructed by adding imperceptibly tiny perturbations to input images (Goodfellow et al., 2014). Unlike traditional graph augmentation techniques such as random edge dropping, which may worsen the model performance (Yu et al., 2022), adding noises to embeddings can preserve more complete graph interaction information, possibly increasing the model’s robustness and effectiveness.

1.3. Report Structure

This report will first introduce basic information of the recommending problems, including variable definitions, related concepts, and the baseline model LightGCL (Cai et al., 2023). Next, the proposed new model structure with a denoising module, cross-layer contrastive loss, and noise to embeddings will be explained, followed by experiments comparing performance of models across various datasets as well as different hyperparameters. After that, the final chapter summarizes the main findings of this research and discusses the future opportunities.

CHAPTER 2

PROBLEM ANALYSIS

2.1. Basic Variables Definition

This project adopts a common approach of user-item recommendation by means of collaborative filtering. Given a user set $\mathbf{U} = \{u\}$ and an item set $\mathbf{V} = \{v\}$, the input user-item interaction matrix can be denoted by \mathbf{R} , which is of size $|\mathbf{U}| * |\mathbf{V}|$. The element in \mathbf{R} , denoted by r_{u_i, v_j} , equals to 1 if there exists an interaction between u_i and v_j , and 0 otherwise. Each user u_i and item v_j will be assigned an embedding vector $e_i^u, e_j^v \in \mathbb{R}^d$, and the embedding size is d . The collections of user and item embeddings are $\mathbf{E}^u \in \mathbb{R}^{|\mathbf{U}|*d}$, $\mathbf{E}^v \in \mathbb{R}^{|\mathbf{V}|*d}$ respectively. In the l -th layer of the graph, the aggregate embeddings for a user u_i will be $z_{i,l}^u$, and for an item v_j will be $z_{j,l}^v$.

2.2. Related Concepts

This section introduces some basic concepts for the problems and techniques related to recommending systems.

Noise in user-item interactions. In recommending tasks, noises are the user-item interactions that fail to reflect user preferences, including natural noise, malicious noise, and other types of noises (Tian et al., 2022). Noise in the input data may make the GNN model perform worse or become more biased.

Collaborative filtering(CF). CF assumes that users with similar behavior would exhibit similar preferences for the same type of items (Wang et al., 2019). Thus, in a traditional CF method, researchers usually describe each user and item with vectorized representations, so that their similarity can be easily computed. After CF reconstructs historical user-item interaction, the predictions between a user and an item can be derived. In this project, we define the preference of user u_i and item v_j as the inner product between their final embeddings: $\hat{y}_{i,j} = (e_i^u)^T e_j^v$, and the final

embeddings can be denoted as the sum of embeddings from each layer: $e_i^u = \sum_{l=0}^L z_{i,l}^u$, $e_j^v = \sum_{l=0}^L z_{j,l}^v$, where L is the total number of layers of GNN (Cai et al., 2023).

Contrastive Graph Learning. There has been a promising research trend in applying graph contrastive learning to recommending systems. Many contrastive learning models, such as SimGCL(Yu et al., 2022) and SGL(Wu et al., 2021), perform data augmentation through random node/edge dropout(Cai et al., 2023). However, this data augmentation method may drop out meaningful interactions, making the data sparsity issue worse(Cai et al., 2023). Thus, instead of performing graph augmentation, this project conducts augmentations to embeddings directly, which is inspired by XSimGCL(Yu et al., 2023). Regarding the loss function, most models adopt a joint learning scheme (Yu et al., 2023), formulated as: $\mathcal{L} = \mathcal{L}_{rec} + \lambda \mathcal{L}_{cl}$, where the recommendation loss $\mathcal{L}_{rec} = -\sum_{(u_i, v_j) \in B} \log(\sigma((e_i^u)^T e_j^v - (e_i^u)^T e_r^v))$. In the recommendation loss, $\sigma()$ is a sigmoid function, and B is the minibatch of user-item interactions, for the current round of loss updating. e_r^v is the embeddings of a randomly sampled item. As for the contrastive loss \mathcal{L}_{cl} , one of the most commonly used loss functions is the InfoNCE loss (Oord et al., 2018), which encourages the consistency between embeddings of positive samples, and minimizes the likelihood between embeddings of negative samples (Yu et al., 2023).

2.3. Baseline Model: LightGCL

LightGCL is an efficient and effective model, by generating a reconstructed graph through Singular Value Decomposition (SVD) (Cai et al., 2023). As shown in *Figure 1.*, By selecting the first q columns with the largest singular values in the SVD result, we get U_q, S_q, V_q^T , and we use these three matrices to reconstruct a normalized adjacency matrix(Cai et al., 2023). This idea originates from the fact that the largest singular values are usually strongly associated with principal components in matrices. Using the original and reconstructed graphs as input, two GCNs aggregate the information(Cai et al., 2023). For a user u_i , the message aggregation process can be

denoted by $z_{i,l}^u = \text{Act}(p(A_{i,:}) \cdot \mathbf{E}_{l-1}^v)$ (Cai et al., 2023). The $\text{Act}()$ is the activation function, which is the identity function, $\text{Act}(x) = x$, and A represents the normalized adjacency matrix, where $A_{i,:}$ should be all neighbors connected to u_i (Cai et al., 2023). The message aggregation process is similar for items. The above aggregation process will also be performed for the reconstructed graph, resulting in learned embeddings denoted by $\mathbf{G}^u \in \mathbb{R}^{|U| \times d}$, $\mathbf{G}^v \in \mathbb{R}^{|V| \times d}$. After that, the contrastive loss will be calculated, which contrasts the embeddings learned from two graphs, denoted by $\mathcal{L}_s^u = \sum_{i=0}^I \sum_{l=0}^L -\log \left(\frac{\exp(\text{CosineSimilarity}(z_{i,l}^u, g_{i,l}^u/t))}{\sum_{i'=0}^I \exp(\text{CosineSimilarity}(z_{i',l}^u, g_{i',l}^u/t))} \right)$, and t is the temperature for contrastive learning (Cai et al., 2023). Besides, the recommending loss will be $\mathcal{L}_r = -\sum_{i=0}^I \sum_{s=1}^S \max(0, 1 - \hat{y}_{i,p_s} + \hat{y}_{i,n_s})$, where the \hat{y}_{i,p_s} and \hat{y}_{i,n_s} are a pair of positive and negative items for user u_i . Thus, the total loss will be $\mathcal{L} = \mathcal{L}_r + \lambda_1 * (\mathcal{L}_s^u + \mathcal{L}_s^v) + \lambda_2 * \|\theta\|_2^2$, where λ_1, λ_2 are regularization weights (Cai et al., 2023).

Despite its high effectiveness and efficiency, LightGCL does not deal with noisy interactions in the original graph, making it less robust under different datasets with various amounts of noise. Moreover, it does not make use of cross-layer contrastive loss, which may be a waste of potential information. Additionally, LightGCL still uses edge dropout to perform graph augmentation, which may also drop useful information. To improve this baseline model, the project proposes methodologies in the following part, to solve each of the problems above.

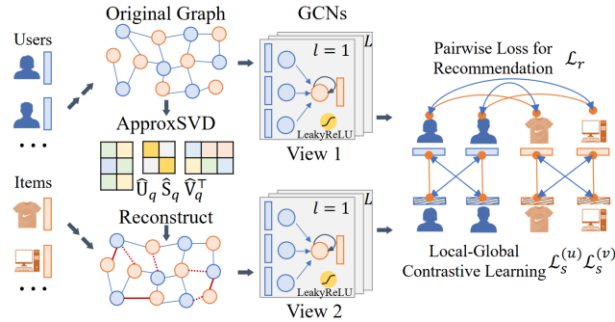


Figure 1. LightGCL Model Structure (Cai et al., 2023)

CHAPTER 3

METHODOLOGY

This project adopts three approaches to improve the baseline model’s performance: the denoising module, adding cross-layer contrastive loss and augment embedding with random noise.

3.1. Denoising Module

Inspired by RGCF(Tian et al., 2022) and AdaGCL (Jiang et al., 2023), we apply a denoising module before every GCN layer. As shown in *Figure 2.*, the denoising module transforms the original user-item interaction matrix \mathbf{R} , which consists of 0s and 1s, into a reliability degree matrix \mathbf{R}_d . We first get the structural features of users and items through $\mathbf{H}_U^s = \mathbf{R}\mathbf{E}^I$ and $\mathbf{H}_I^s = \mathbf{R}^T\mathbf{E}^U$. For interaction between user u_i and item v_j in \mathbf{R} , we compute the normalized cosine distance as the reliability degree,

which is $s_{u_i, v_j} = \frac{h_i^u \cdot h_j^v}{\|h_i^u\|_2 \|h_j^v\|_2} + 1$ (Tian et al., 2022). s_{u_i, v_j} is regarded as the weight of the

edge in the graph, and the elements in \mathbf{R}_d will be $r_{u_i, v_j} = \mathbb{I}(s_{u_i, v_j} > \beta) * s_{u_i, v_j}$, where $\mathbb{I}(x) = 1$ if $x = True$, and $\mathbb{I}(x) = 0$ if $x = False$ (Tian et al., 2022). This indicates that if the reliability of an interaction is lower than a threshold β , we drop this edge directly. Next, we feed the normalized \mathbf{R}_d to the GCN layers. We apply this denoising module before every GCN layer, as shown in *Figure 3*.

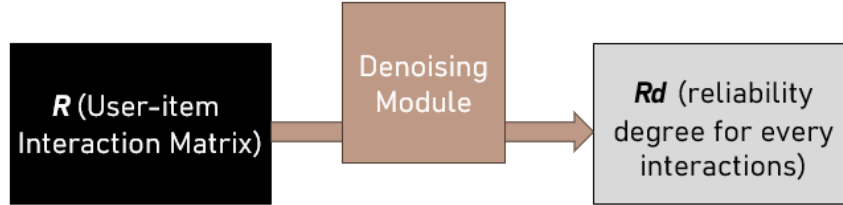


Figure 2. The denoising module

Despite its computational cost, this denoising method assigns every interaction with structural similarity information (Tian et al., 2022). Instead of treating every interaction as the same value 1 in \mathbf{R} , this project use \mathbf{R}_d , where a high r_{u_i, v_j} implies high reliability of interactions. This graph augmentation may be more robust, since it alleviates the impact of noisy interactions by assigning a low reliability (Tian et al., 2022), and preserves different reliability scores for each reliable interaction.

It is worth mentioning that this project uses Pytorch to perform graph normalization faster than the original LightGCL code. Initially, the LightGCL code normalizes every interaction one by one, which is quite time-consuming even for normalizing the adjacency matrix once. This project transforms the original adjacency matrix to Pytorch tensor, so that GPU can speed up the normalization process, and enable efficient normalization after every denoise module.

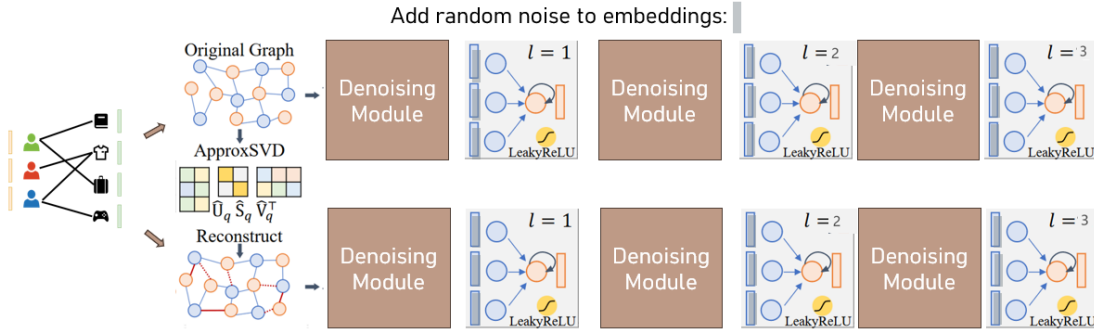


Figure 3. Improved structure of denoising module and GNN based on LightGCL

3.2. Cross-Layer Contrastive Loss

To utilize the information across different layers in GCN model, this project proposes a method for computing cross-layer contrastive loss. After L layers of GCNs, we calculate the average user and item embeddings through $\mathbf{E}^{Uavg} = \frac{\mathbf{E}^U}{L+1}$, $\mathbf{E}^{Iavg} = \frac{\mathbf{E}^I}{L+1}$, where \mathbf{E}^U and \mathbf{E}^I becomes the sum of embeddings for all layers after message aggregation, as mentioned in the Collaborative Filtering part in **Chapter 2.2**. We compute the contrastive loss between the first layer embeddings and the average

embeddings after the final layer, because according to the experiments (Yu et al., 2023), the model performance is higher when contrasting these two layer embeddings than utilizing other intermediate layers. The cross-layer contrastive loss for users is defined as $\mathcal{L}_{cl}^u = \sum_{i=0}^l -\log \left(\frac{\exp(\text{CosineSimilarity}(z_{i,1}^u, e_i^{uavg}/t))}{\sum_{i'=0}^l \exp(\text{CosineSimilarity}(z_{i,1}^u, e_{i'}^{uavg}/t))} \right)$, where $e_i^{uavg} \in \mathbf{E}^{uavg}$. The cross-layer contrastive loss for items is similarly defined. Therefore, in this project, the total loss function will be defined as $\mathcal{L} = \mathcal{L}_r + \lambda_1 * ((1 - w) * (\mathcal{L}_s^u + \mathcal{L}_s^v) + w * (\mathcal{L}_{cl}^u + \mathcal{L}_{cl}^v)) + \lambda_2 * ||\theta||_2^2$, where w is the percentage of cross-layer contrastive learning loss taken in the contrastive loss, and $||\theta||_2^2$ is the $L2$ regularization term.

The cross-layer contrastive loss can enhance the efficacy of graph contrastive learning, by utilizing the high-frequency information in representation-level augmentations (Yu et al., 2023). Through the experiments, we also discover that the cross-layer contrastive loss can speed up the model convergence, making the model to reach a high recall and NDCG with much less number of epochs than the original model.

3.3. Add Noises to Embeddings

This project also adds random, small-value noises to embeddings directly, for the purpose of performing embedding augmentation. It aims at generating small deviation by rotating embeddings at a very small angle in the embedding space, so that a more general characteristic for each users and items can be learned (Yu et al., 2022). We let the embeddings for u_i at layer l to be $z'_{i,l}^u = z_{i,l}^u + ((\epsilon * r) \odot \text{sign}(z_{i,l}^u))$, where the noise vector $r \in \mathbb{R}^d \sim \text{Uniform}(0,1)$. ϵ is used for adjusting the magnitude of the noise value, making the noise numerically equivalent to points on a hypersphere with radius equals to ϵ . *Figure4* shows an example of process of noise adding in a 2-dimension embedding space, in which the θ_1 is the angle that the user embedding rotated, and $z_{j,l}^u$ represents another user's embedding different from $z_{i,l}^u$ (Yu et al., 2022). \odot is the element-wise product, and $\text{sign}()$ is the sign function.

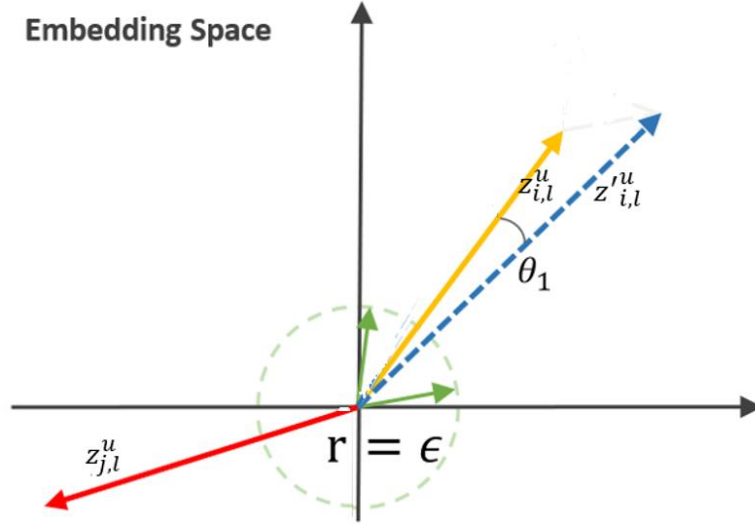


Figure 4. Adding noise in a 2-dimension embedding space (Yu et al., 2022)

Therefore, the complete model structure for our project is shown in Figure 5.

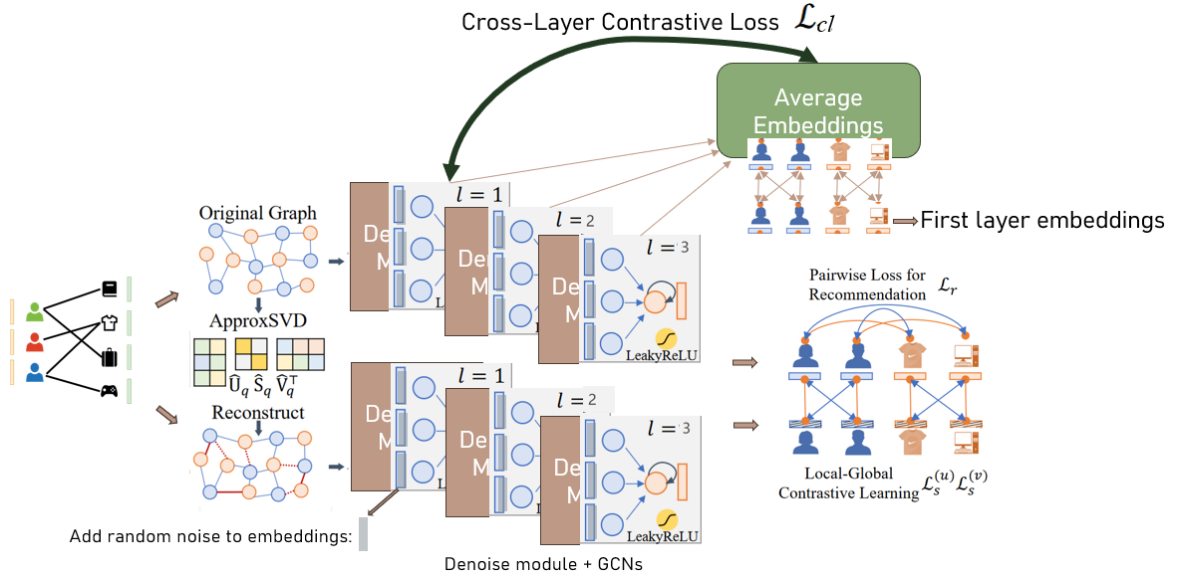


Figure 5. The complete model structure

CHAPTER 4

EXPERIMENTAL ANALYSIS

4.1. Experiment Settings

This project evaluates the model performance compared to the original LightGCL model. We adopt two evaluation metrics: recall@N, which indicates the percentage of the total number of relevant items in the top-N recommending results; Normalized Discounted Cumulative Gain (ndcg@N), measuring the ranking quality. $N=\{20,40\}$. We evaluate the models on three different real-world datasets: **Yelp** (29,601 users, 24,734 items, 1,069,128 interactions): a dataset containing rating interactions on Yelp platform; **Gowalla** (50,821 users, 57,440 items, 1,172,425 interactions): a dataset recording users’ check-in information on Gowalla platform; **Tmall** (47,939 users, 41,390 items, 2,357,450 interactions): an E-commerce dataset including users’ purchase records on various items in Tmall online-shopping platform.

4.2. Hyperparameter Analysis

We tune the hyperparameters using the Yelp dataset. To perform a fair comparison for different values of hyperparameters, we adopt the following fixed settings from the LightGCL original code (Cai et al., 2023): temperature t as 0.2, the regularization weights $\lambda_1=0.2$ and $\lambda_2=1e-7$, the rank for SVD as $q=5$. The embedding size $d=64$, and batch size is set as 256. The number of layers for GCN is set as $L=2$. As for the newly added hyperparameters β , and ϵ , we adopt most values in the range from the original papers (Yu et al., 2022; Yu et al., 2023; Tian et al., 2022), and also added some value outside the ranges, which are underlined, to explore the impact of the hyperparameters further: $\beta = \{0.02, 0.05, 0.1, \underline{0.2}\}$, $\epsilon = \{0.01, 0.025, 0.05, 0.1\}$. w is in the range of $\{0.05, 0.1, 0.2, 0.3, 0.5\}$.

As shown in *Table 1*, when the cross-layer contrastive loss takes weight at $w=0.1$, the model has a slight performance improvement. In *Figure 6*, the performance

drops when w is too high, possibly due to the information quality of cross-layer contrastive learning not being as high as the contrastive view between original and reconstructed graph embeddings. Thus, the model training process may not depend too much on the cross-layer contrastive loss. Meanwhile, the experiments witnessed performance improvement when the denoising module was adopted, especially when $\beta=0.05$. This improvement may indicate that the denoising module can eliminate part of noisy interactions, making the project model more effective. The experiments also show that adding random noise to embeddings may not significantly improve the model’s performance.

	recall@20	ndcg@20	recall@40	ndcg@40
LightGCL	0.1004	0.0863	0.1585	0.1075

CrossLayer_weight $w=$	recall@20	ndcg@20	recall@40	ndcg@40
0.05	0.1001	0.0862	0.1598	0.1080
0.1	0.1005	0.0869	0.1593	0.1082
0.2	0.1005	0.0864	0.1594	0.1078
0.3	0.0998	0.0857	0.1582	0.1069
0.5	0.0984	0.0842	0.1571	0.1054

Improvement			
-0.25%	-0.08%	0.78%	0.45%
0.12%	0.67%	0.47%	0.64%
0.08%	0.09%	0.54%	0.23%
-0.60%	-0.64%	-0.20%	-0.57%
-2.02%	-2.45%	-0.92%	-1.94%

Denoise_beta $\beta=$	recall@20	ndcg@20	recall@40	ndcg@40
0.02	0.1009	0.0871	0.1595	0.1086
0.05	0.1019	0.0881	0.1626	0.1102
0.1	0.1004	0.0864	0.1595	0.108
0.2	0.1015	0.0872	0.1614	0.1089

Improvement			
0.52%	0.94%	0.58%	0.98%
1.48%	2.16%	2.57%	2.51%
0.01%	0.10%	0.62%	0.47%
1.09%	1.02%	1.79%	1.33%

Embed_noise_eps $\epsilon=$	recall@20	ndcg@20	recall@40	ndcg@40
0.01	0.0991	0.0855	0.1573	0.1067
0.025	0.1001	0.0863	0.1603	0.1081
0.05	0.0984	0.0846	0.1569	0.1059
0.1	0.0997	0.086	0.1599	0.1078

Improvement			
-1.23%	-0.87%	-0.76%	-0.71%
-0.27%	0.00%	1.13%	0.60%
-1.98%	-1.94%	-1.01%	-1.45%
-0.70%	-0.30%	0.82%	0.28%

Table 1. Hyperparameter Tuning for cross-layer contrastive learning weight w , the denoise threshold β , and the ϵ in noise magnitude adjustment.

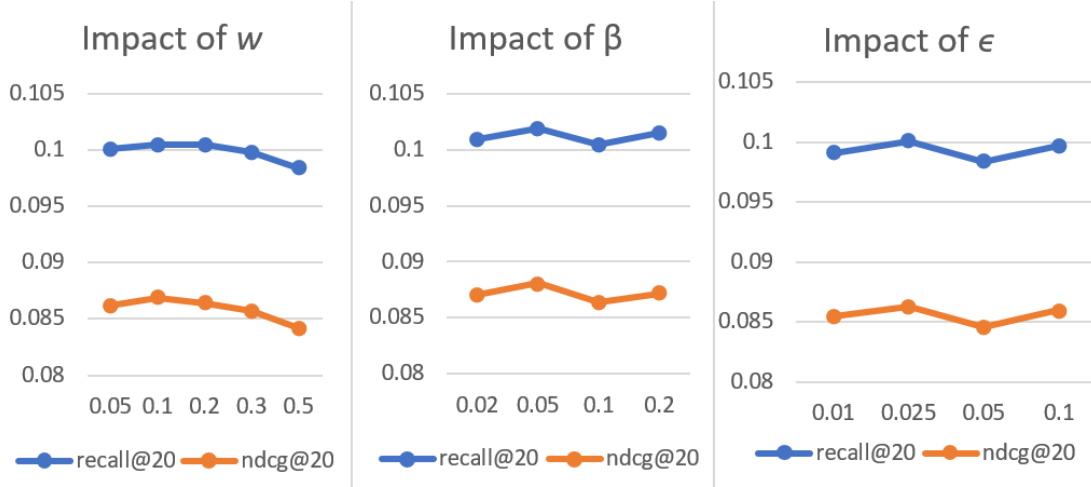


Figure 6. The impact of different value of w , β , and ϵ on model performance

4.3. Comparative Analysis

This project adopts different epochs across datasets, because the model reaches optimal at different number of epochs under various datasets. As a result, the experiments perform early stopping when the performance drops after several epochs, to reduce the training time and avoid overfitting. Through several trials of experiments, we set the number of epochs for both models' experiments as: Yelp(100 epochs), Gowalla(50 epochs), and Tmall(30 epochs).

The experiment results across different datasets are shown in *Table 2*. We utilized the tuned hyperparameters for our model, which are shown in bold in **Chapter 4.2.** During the experiment, we also tried different hyperparameters for different datasets. We selected the hyperparameters for different datasets that achieved the best performance as: Yelp($\beta = 0.05$, $\epsilon = 0.025$, $w = 0.1$), Gowalla($\beta = 0.1$, $\epsilon = 0.025$, $w = 0.2$), Tmall($\beta = 0.1$, $\epsilon = 0.025$, $w = 0.2$) We can conclude that our model reaches a better performance than LightGCL across different datasets.

Dataset: Yelp	recall@20	ndcg@20	recall@40	ndcg@40
LightGCL	0.1004	0.0863	0.1585	0.1075
Project Model	0.1020	0.0877	0.1617	0.1095
Improvement	1.56%	1.64%	1.99%	1.87%

Dataset: Gowalla	recall@20	ndcg@20	recall@40	ndcg@40
LightGCL	0.2086	0.1218	0.2964	0.1446
Project Model	0.2223	0.1324	0.3083	0.1549
Improvement	6.56%	8.74%	4.02%	7.11%

Dataset: Tmall	recall@20	ndcg@20	recall@40	ndcg@40
LightGCL	0.0843	0.0602	0.1295	0.0759
Project Model	0.0863	0.0615	0.1322	0.0774
Improvement	2.38%	2.24%	2.15%	2.07%

Table 2. Performance comparison with LightGCL on Yelp, Gowalla and Tmall

It is worth noting that the project model reaches a high recall using much fewer epochs than the LightGCL model, as displayed in *Figure 7*, especially in the Yelp dataset. This may be due to two reasons: (1) The denoising module provides interaction data with less noise, making the model trained with less noisy interactions, and reach a high recall faster. (2) Adding cross-layer contrastive loss may provide the model with more information to learn, so that it converges to the optimal faster than LightGCL.

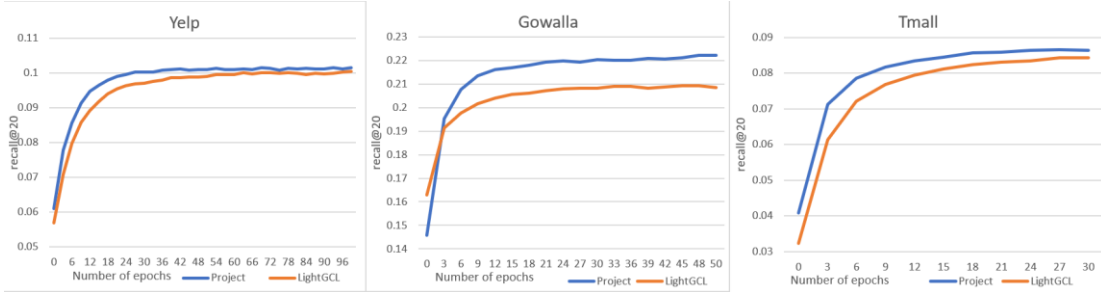


Figure 7. recall@20 with number of epochs of project model and LightGCL

CHAPTER 5

CONCLUSION

This project proposes a recommending system based on the LightGCL model, with improvement by utilizing denoising modules to lessen the influence of noisy interactions, increasing the model’s robustness across different datasets with different amount of noise. Additionally, the model extracts information across different GNN layers by taking cross-layer contrastive loss into account. The experiments show that random noises added to embeddings may be ineffective.

Despite the improvements, this project has the following limitations. Firstly, the efficiency of for our model remains to be improved. Compared with LightGCL, adding the denoising function as well as cross-layer contrastive loss leads to high computational cost, with approximately 30% more time in the training process using the Yelp dataset. We may improve the efficiency by reducing the number of denoising modules or adopting loss functions that require less computation. The second limitation for this project is that more datasets can be utilized in the experiments to further prove the robustness of our model. Third, this project does not provide a more effective augmentation method, since the experiments showed that adding noises to embeddings directly has very slight performance improvement.

As for future research, we may further discover effective and efficient methods to eliminate noisy interactions. Moreover, more contrastive learning frameworks and effective augmenting methods remain to be explored. As more and more Large Language Models emerge, we may also leverage the LLM’s ability to analyze and detect noisy user-item interactions. LLMs may also be helpful for graph augmentation through their ability to understand item node attributes and user profiles(Wei et al., 2023).

REFERENCES

- Cai, X., Huang, C., Xia, L., & Ren, X. (2023). LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. arXiv preprint arXiv:2302.08191.
- Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.
- Jiang, Y., Huang, C., & Huang, L. (2023). Adaptive graph contrastive learning for recommendation. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 4252-4261).
- Oord, A. V. D., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748.
- Tian, C., Xie, Y., Li, Y., Yang, N., & Zhao, W. X. (2022). Learning to denoise unreliable interactions for graph collaborative filtering. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 122-132).
- Wang, J., Louca, R., Hu, D., Cellier, C., Caverlee, J., & Hong, L. (2020). Time to Shop for Valentine's Day: Shopping Occasions and Sequential Recommendation in E-commerce. In Proceedings of the 13th International Conference on Web Search and Data Mining (pp. 645-653).
- Wang, X., He, X., Wang, M., Feng, F., & Chua, T. S. (2019). Neural graph collaborative filtering. In Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval (pp. 165-174).
- Wei, W., Ren, X., Tang, J., Wang, Q., Su, L., Cheng, S., ... & Huang, C. (2023). LLMRec: Large Language Models with Graph Augmentation for Recommendation. arXiv preprint arXiv:2311.00423.
- Wei, Y., Wang, X., Nie, L., He, X., Hong, R., & Chua, T. S. (2019). MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video. In Proceedings of the 27th ACM international conference on multimedia (pp. 1437-1445).
- Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., & Xie, X. (2021). Self-supervised graph learning for recommendation. In Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval (pp. 726-735).
- Yang, B., Liu, D., Suzumura, T., Dong, R., & Li, I. (2023). 🌟 Going Beyond Local: Global Graph-Enhanced Personalized News Recommendations. In Proceedings of the 17th ACM Conference on Recommender Systems (pp. 24-34).

- Yu, J., Xia, X., Chen, T., Cui, L., Hung, N. Q. V., & Yin, H. (2023). XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering*.
- Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., & Nguyen, Q. V. H. (2022). Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 1294-1303).
- Zhang, H., Tian, C., Li, Y., Su, L., Yang, N., Zhao, W. X., & Gao, J. (2021, August). Data poisoning attack against recommender system using incomplete and perturbed data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (pp. 2154-2164).