Large Language Model augmented Positive Negative Graph Neural Network for Recommendation

Li Haowei
HKUST
hliei@connect.ust.hk

Kang Zhaowei
HKUST
zkang@connect.ust.hk

Tang Kewei
HKUST
ktangaj@connect.ust.hk

Abstract

Recommendation systems are crucial in today's digital age for personalizing user experiences and driving engagement. Therefore, enhancing the performance of recommendation systems is essential for maximizing their impact. We proposed a recommendation system that fully utilizes dataset side information and positive/ negative feedback to produce accurate and relevant content recommendations. By leveraging LLM's powerful reasoning ability to augment the dataset and taking two kinds of feedback into account, our model produces more accurate results than the baseline model. We also evaluate our model's performance on a real collected dataset.

1. Introduction

Recommendation systems are widely used in various applications today. Many technology enterprises adopt recommendation algorithms in fields such as news, shorts, and movies. The recommendation system learns user's preferences from historical data and gives out potential items that users may be interested in.

One common approach to building a recommendation system is to model the user-item interaction by a Graph neural network(GNN). GNN models the user and items as points and the user-item interaction as edges. By learning from the graph structure, GNN generated an embedding for every user and item and predicted the user's interest based on the embeddings.

Problem 1. GNN-based recommendation systems generally leverage the topology of user-item interaction graphs to generate embeddings and make predictions. They primarily focus on the graph structure while often overlooking the rich side information available. Although effective, these methods often suffer from the challenge of data sparsity [8], where the lack of sufficient user-item interactions severely limits the model's ability to learn meaningful representations. This issue is particularly prominent in real-world recommendation scenarios, where many users and items have limited or no interactions in the graph.

Problem 2. GNN is generally trained under a network homophily assumption that a target node and its neighbors are similar to each other [2]. However, in the context of the recommendation system, the actual graph for learning is the rating bipartite graph which contains both positive and negative feedback. Popular GNN-based recommendation systems eg. PinSage [5], NGCF [9], LightGCN [1] do not incorporate the negative feedback information in their message-passing of the user-item bipartite graph, which limits their scalability.

Our team has built a novel system called LLM-PNGNN to offer a new method for user-item embeddings. Firstly, recognizing that data sparsity can lead to prediction errors, we incorporate side information by large language models(LLMs) for data augmentation. To fully leverage user feedback, our model transmits messages through a bipartite graph, which is then categorized into positive and negative feedback graphs. An attention mechanism is incorporated to combine embeddings from two graphs, generating the final embedding for prediction. Besides, we leverage a new loss function built upon standard BPR Loss, called Dual-BPR Loss, which effectively incorporates two kinds of feedback in the optimization stage. In our experiments, we have evaluated our model performance by using Precision@K and Recall@K metrics. In the experiment, we compare the model with the baseline model LightGCN. Furthermore, we perform the ablation study to validate the effect of LLM augmentation on the model training through the comparison between the model trained by the augmented dataset and the model trained by the raw dataset as well as the effect of our positive negative partition architecture. The results demonstrate the effectiveness of large language augmentation in enhancing training datasets and highlighting the impact of our new GNN architecture on improving recommendation performance.

The rest of this report is organized as follows: Sec. 2 introduces our model's architecture in detail. Sec. 3 evaluates our model's performance on the MovieLen dataset. Sec. 4

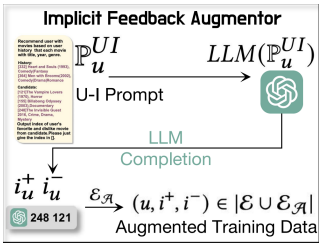

Figure 1. The process of our dataset augmentation using LLM

gives a conclusion to our model.

2. METHODOLOGY

Our model consists of two parts: The LLM argumentation and the PN-GNN. This section will explain the architecture of our model in detail.

2.1. LLM Graph Augmentation

The data sparsity issue plagued recommendation systems for a long time, and previous studies attempted to mitigate this problem by incorporating side information [8]. However, these approaches lead to serious side effects such as noise and low data quality while utilizing side information. With advancements in large language models, we can now more efficiently extract the semantic meaning from side information. This allows us to leverage the side information with minimal side effects. In our model, we utilize the side information to augment the user-item interaction.

We lead in LLM to help our model fully utilize the side information in the dataset. Figure1 shows our dataset augmentation process. In detail, we refer to the newly published GNN model LLMRec [8], we employed a similar approach to enhance our training dataset for improving performance in our graph neural network project. Specifically, our methodology consists of the following steps: First, we utilized a pre-trained LightGCN model to generate initial embeddings and corresponding recommendations. Next, we sampled the user's historical rating records and the recommendations generated to create input prompts for the LLM. The LLM processed these prompts and outputted the suggested augmented edges. Finally, we augmented the training dataset based on these suggested edges, thereby enriching the data and potentially improving

2.2. PN-GNN architecture

The architecture of the PN-GNN model is depicted in Fig. 2, which consists of three main parts:

- Positive Negative Partition for the ratings bipartite graph.
- Embedding generation for each partitioned graph.
- Optimization via a Dual-BPR loss.

2.2.1 Positive Negative Partition

First, we set the rating threshold to separate positive and negative feedback. We use 3.5 in our case to separate the raw rating graph into Positive graph $\boldsymbol{\mathcal{G}p}$ and Negative graph $\boldsymbol{\mathcal{G}n}$.

GNN models are trained by aggregating information under the homophily assumption [2] that a target node and its neighbors are similar to each other. However, since for the recommendation system with raw rating graph $\mathcal{G}$ that includes negative edges (i.e. interactions with low ratings), aggregating the information of such negatively connected neighbors may not be desirable. Solely training on all scale ratings seems to overlook negative feedback.

To fully utilize both positive and negative ratings, we partition the bipartite graph [3] to earn both positive and negative feedback bipartite graphs (ie. $\mathcal{G}p$ and $\mathcal{G}n$) and learn embeddings in these 2 graphs. This can ensure that the homophily assumption is met in both graphs.

2.2.2 Message Passing on $\mathcal{G}p$ and $\mathcal{G}n$

Second, we will describe how to generate embeddings of users and items along with two learning models after obtaining the Positive graph $\mathcal{G}p$ and Negative graph $\mathcal{G}n$.

To effectively aggregate and propagate these embeddings, PN-GNN utilizes a technique called LightGCN [1], which has emerged as an effective simplified GCN method on the recommendation system by simplifying the diffusion propagation process to directly encourage the embeddings of similar users/items to be similar.

In the message passing process on the positive graph, the embeddings $z_u^{(k+1)}$ and $z_i^{(k+1)}$ at the (k+1)-th layer are updated by summing the normalized embeddings at the k-th layer:

$$z_u^{(k+1)} = \sum_{i \in \mathcal{N}_p(u)} \frac{1}{\sqrt{|\mathcal{N}_p(u)|}\sqrt{|\mathcal{N}_p(i)|}} z_i^{(k)}, \quad (1)$$

$$z_i^{(k+1)} = \sum_{u \in \mathcal{N}_p(i)} \frac{1}{\sqrt{|\mathcal{N}_p(i)|}\sqrt{|\mathcal{N}_p(u)|}} z_u^{(k)}. \quad (2)$$

where $N_p(u)$ and $N_p(i)$ are the set of the neighbors of node u and node i in the positive graph. The final embeddings z_u and z_i derived by positive graph can be obtained by averaging the interest embeddings from all layers:

$$z_u = \frac{1}{K+1} \sum_{k=0}^{K} z_u^{(k)}, \quad z_i = \frac{1}{K+1} \sum_{k=0}^{K} z_i^{(k)}. \quad (3)$$

In Eqs. (1) to (3), the initial embeddings $z_u^{(0)}$ and $z_i^{(0)}$ are initialized randomly as the trainable parameters which will be used later in loss calculation. Now we can integrate the

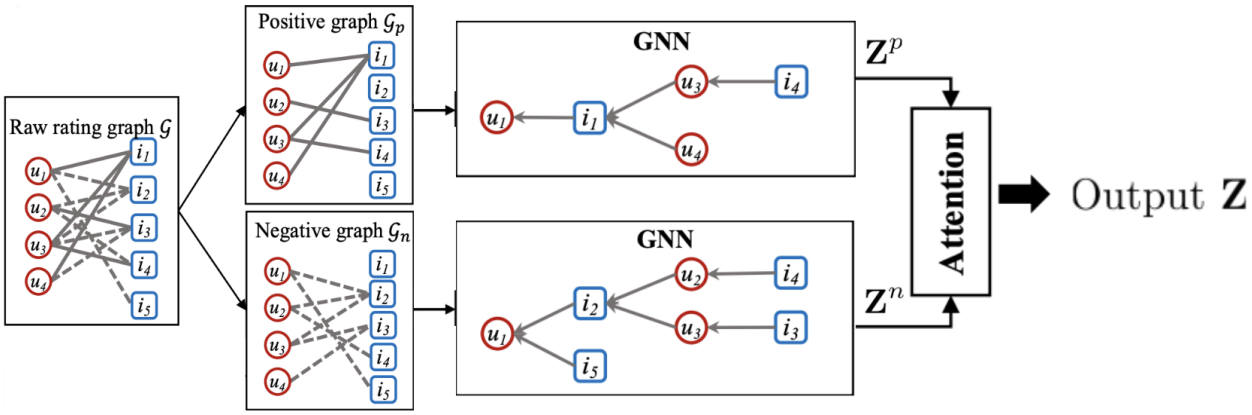

Figure 2. The overview architecture of PN-GNN.

vector form into matrix representation:

$$\boldsymbol{Z^p} = \frac{1}{K+1}\sum_{k=0}^{K} Z^{(k)}, \quad Z^{(k+1)} = \left(D_p^{-\frac{1}{2}} A_p D_p^{\frac{1}{2}}\right) Z^{(k)}. \tag{4}$$

where $\boldsymbol{D_p}$ is the degree matrix and $\boldsymbol{A_p}$ is the adjacency matrix of $\mathcal{G}p$.

The above process can be also adopted for the negative graph $\mathcal{G}n$ which will first randomly initialize the learnable embeddings $v_u^{(0)}$ and $v_i^{(0)}$, then generate the final embeddings v_u and v_i through the same LightGCN-type diffusion propagation process. Finally, we obtain 2 matrices $\boldsymbol{Z^p}$ and $\boldsymbol{Z^n}$ representing all the users' and items' embeddings correspondingly from positive and negative graphs.

Next, to determine the importance of the embeddings $\boldsymbol{Z^p}$ and $\boldsymbol{Z^n}$ in generating the final embedding $\boldsymbol{Z}$, we employ an attention mechanism. [7]

$$(\alpha^p, \alpha^n) = \text{ATTENTION}(\boldsymbol{Z^p}, \boldsymbol{Z^n}), \tag{5}$$

which results in the final embeddings:

$$\boldsymbol{Z} = (\alpha^p \mathbf{1}_{\text{attn}}) \odot \boldsymbol{Z^p} + (\alpha^n \mathbf{1}_{\text{attn}}) \odot \boldsymbol{Z^n}. \tag{6}$$

2.2.3 Optimization

Third, we explain the optimization of the PN-GNN method in the training phase via Dual-BPR Loss [3].

We denote the training set for the whole graph $\mathcal{D}$ as:

$$\mathcal{D} = \left\{(u, i, j_n) \mid j_n \notin \mathcal{N}(u), n \in \{1, \cdots, N_{neg}\}\right\}, \tag{7}$$

where N_{neg} denotes the number of negative samples, here we adopt a method called Degree-based Negative Sampling. In this method, we first obtain a degree-based noise distribution of the items $P(j) \propto d_j^{3/4}$ [4] where d_j is the degree of item j and the power of 0.75 for smoothing the distribution. Then we do negative sampling based on this distribution which mirrors real-world scenarios where popular items are more frequently encountered by users.

The we can split the training set $\mathcal{D}$ for loss calculation in 2 sets as $\mathcal{D}_p$ and $\mathcal{D}_n$ as:

$$\mathcal{D}_p = \left\{(u, i, j_n) \mid (u, i, j_n) \in \mathcal{D}, (u, i) \in \mathcal{G}p\right\}, \tag{8}$$

$$\mathcal{D}_n = \left\{(u, i, j_n) \mid (u, i, j_n) \in \mathcal{D}, (u, i) \in \mathcal{G}n\right\}. \tag{9}$$

Furthermore, we leverage mini-batch learning to train our model. We denote the batch set as $\mathcal{B}$ and each batch of partitions as $\mathcal{B}_p \subset \mathcal{D}_p$ and $\mathcal{B}_n \subset \mathcal{D}_n$, respectively. And we denote the initial embeddings of the 0-th layer for a given epoch as $\mathbf{E}^{(0)}$, which contains all the embeddings of the user's, items, and negative sample items $(u, i, j) \in \mathcal{B}$. Then now we can present the Dual-BPR Loss function, which is built upon the original BPR Loss function [6] that is widely used in the recommender system :

$$\begin{aligned}
\mathcal{L}_{\text{DB}} = &- \sum_{(u,i,j)\in\mathcal{B}_p} \ln \sigma(\hat{y}_{u,i} - \hat{y}_{u,j}) - \\
&\sum_{(u,i,j)\in\mathcal{B}_n} \ln \sigma(\hat{y}_{u,j} - \hat{y}_{u,i}) + \lambda_{\text{reg}} \|\boldsymbol{E}^{(0)}\|^2, \\
\hat{y}_{u,i} &= \begin{cases} \boldsymbol{z}_u \boldsymbol{z}_i^\top, & \text{if } (u,i,j) \in \mathcal{B}_p \\ b \cdot \boldsymbol{v}_u \boldsymbol{v}_i^\top, & \text{if } (u,i,j) \in \mathcal{B}_n \end{cases} \\
\hat{y}_{u,j} &= \begin{cases} \boldsymbol{z}_u \boldsymbol{z}_j^\top, & \text{if } (u,i,j) \in \mathcal{B}_p \\ \boldsymbol{v}_u \boldsymbol{v}_j^\top, & \text{if } (u,i,j) \in \mathcal{B}_n \end{cases}
\end{aligned} \tag{10}$$

where λ_{reg} is a hyperparameter that controls the L2 regularization strength; $\sigma(x)$ is the sigmoid function and $b > 1$ is the feedback aware coefficient.

As shown in Eq. (10), the only difference between Dual-BPR Loss and original BPR loss is that it introduces a coefficient b. The Dual-BPR Loss function accounts for two

Table 1. Summary of Experiment Data

Method	K=5		K=10		K=15		K=20	
	Precision@K	Recall@K	Precision@K	Recall@K	Precision@K	Recall@K	Precision@K	Recall@K
LightGCN	.2772	.1070	.2330	.1721	.2052	.2200	.1855	.2571
LLM-LightGCN	.2903	.1124	.2396	.1825	.2089	.2218	.1870	.2568
PNGNN	.3164	.1165	.2687	.1871	.2383	.2499	.2151	.2915
LLM-PNGNN	**.3288**	**.1260**	**.2737**	**.1978**	**.2412**	**.2521**	**.2171**	**.2930**

types of observed items, which include both positive and negative relations between users and items, as well as unobserved ones. By introducing the coefficient b, the predicted preference for an observed item is higher than its unobserved counterparts along with the induced difference between high and low ratings and ensures the following priority order: positive feedback $>$ negative feedback $>$ no feedback.

3. Experiment

In this section, we will introduce the dataset we used for the model training, along with the model performance evaluation method and the corresponding evaluation result. In addition, we will compare LLM-PNGNN with several models, including LightGCN (Baseline), PNGNN (without LLM augmentation), and LLM-LightGCN (with LLM augmentation).

3.1. Dataset

MovieLen is a well-known benchmark dataset containing millions of ratings from the past few decades. To save time and reduce computation resource consumption in the experiment, we adopted a relatively moderate size and considered stable dataset called MovieLen-1M to train and test our model, which comprises one million ratings from about 6000 users on 4000 thousand movies.

3.2. Evaluation Metric

We evaluate the effectiveness of LLM-PNGNN by using Precison@K and Recall@K, two common metrics for evaluating the ranking algorithm performance. The Precision@K can be expressed as:

$$\text{Precision@}K = \frac{1}{|U|}\sum_{u\in U}\frac{|GT_u \cap R_u(K)|}{K}, \qquad (11)$$

where U is the user set, GT_u is the ground truth item set liked by user u, and $R_u(K)$ denotes the recommended top-K items for user u. The Recall@K can be expressed as:

$$\text{Recall@}K = \frac{1}{|U|}\sum_{u\in U}\frac{|GT_u \cap R_u(K)|}{|GT_u|}, \qquad (12)$$

Precision@K captures the proportion of relevant items among the top-K recommended results for a user and Recall@K reflects the proportion of relevant items among all correct results for a user. Both of them provide us an insight into the accuracy and quality of a ranking model.

3.3. Experiment Setting

The hyperparameter setting is in Tab. 2. In the experiments, we set the embedding to 64. We use the Adam optimizer with a learning rate of 5e-3 to optimize our model. Mini-batch learning is used in the training process, and the batch size is set to 4096. We train the model for a total of 60 epochs. We employ L2 regularization with a coefficient of 0.05 to avoid over-fitting. Negative sampling is incorporated during training, and we set the number of negative samples to 40. The architecture of LLM-PNGNN contains 4 layers of GNNs. The filter rating threshold adopts 3.5 to distinguish like or unlike.

Parameter	**Value**
Optimizer	Adam
Batch Size	4096
Embedding Dimension	64
Learning Rate	5×10^{-3}
Regulation Coefficient	0.05
Threshold	3.5
Num Neg Samples	40
GNN Layers	4
Epoch	60

Table 2. Training Parameters

3.4. Experiment result

In the experiment, we split the dataset for training and testing. The training set contains 80% of the ratings, while the remaining 20% constitutes the test set. We perform split three times and earn 3 sets of training and testing pairs. The final result will be calculated by the average of results from these 3 independent experiments. The experiment result is shown in Tab. 1, and the best result in each column is bolded. This table presents a performance result for LLM-PNGNN and LightGCN using the evaluation metrics, and with varying values of K. For better comparison, we further

add the experiment data for the ablation study, in which we control the variable: partition GNN architecture, and variable LLM augmentation to give a comprehensive evaluation for our model.

3.4.1 Overall performance

Our model LLM-PNGNN by comparing with the baseline LightGCN, shows its quality, by achieving a distinguishing higher accuracy in the scenery when considering negative feedback. It demonstrates the scalability of LLM-PNGNN in the scenery considering the negative feedback.

3.4.2 Ablation study

Variable–LLM Augmentation. We notice that, after being trained with the dataset augmented by LLM, both LightGCN and PN-GNN gain visible improvement in prediction accuracy, which supports that the LLM augmentation is an efficient approach to mitigate the impact data sparsity and leverage the side information to enhance the training model performance.

Variable–Positive Negative Partition Architecture. Besides, without the partition GNN architecture, LightGCN shows worse performance than PN-GNN in the experiment, indicating that the partition graph architecture will benefit the recommendation accuracy when needing to consider the effect of negative feedback.

4. Conclusion

In this report, we developed a Graph Neural Network (GNN) model that excels in leveraging large language models (LLMs) for data augmentation, incorporating both positive and negative feedback within the recommendation system. The result is a significant performance improvement. Our findings demonstrate that fully integrating diverse information sources, including both types of feedback and side information, can enhance model performance. We believe that a deeper integration of LLMs with GNNs holds the potential for even greater advancements in various GNN applications.

References

[1] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. 2020. 1, 2

[2] L. Zhao M. Heimann L. Akoglu J. Zhu, Y. Yan and D. Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. 2022. 1, 2

[3] Wang C. Xu J. Wu C. Zheng K. Song Y. Mou N. Liu, Z. and K. Gai. Pane-gnn: Unifying positive and negative edges in graph neural networks for recommendation. 2023. 2, 3

[4] Sutskever I. Chen K. Corrado G. Mikolov, T. and J. Dean. Distributed representations of words and phrases and their compositionality. 2013. 3

[5] Kaifeng Chen Pong Eksombatchai William L. Hamilton Rex Ying, Ruining He and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. 2018. 1

[6] Zeno Gantner Steffen Rendle, Christoph Freudenthaler and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. 2009. 3

[7] Shazeer N. Parmar N. Uszkoreit J. Jones L. Gomez A. N. Kaiser L. Vaswani, A. and I. Polosukhin. Attention is all you need. 2017. 3

[8] Tang J et al Wei W, Ren X. Llmrec: Large language models with graph augmentation for recommendation. *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 2024. 1, 2

[9] Meng Wang Fuli Feng Xiang Wang, Xiangnan He and Tat-Seng Chua. Neural graph collaborative filtering. 2019. 1