



Position-aware Graph Transformer for Recommendation

JIAJIA CHEN, Institute of Dataspace, Hefei Comprehensive National Science Center, Hefei, China

JIANCAN WU, MoE Key Laboratory of Brain-Inspired Intelligent Perception and Cognition,

University of Science and Technology of China, Hefei, China

JIawei CHEN, School of Computer Science and Technology, Zhejiang University, Hangzhou, China

CHONGMING GAO, MoE Key Laboratory of Brain-Inspired Intelligent Perception and Cognition,

University of Science and Technology of China, Hefei, China

YONG LI, Department of Electronic Engineering, Tsinghua University, Beijing, China

XIANG WANG, MoE Key Laboratory of Brain-Inspired Intelligent Perception and Cognition,

University of Science and Technology of China, Hefei, China

Collaborative recommendation fundamentally involves learning high-quality user and item representations from interaction data. Recently, graph convolution networks (GCNs) have advanced the field by utilizing high-order connectivity patterns in interaction graphs, as evidenced by state-of-the-art methods like PinSage and LightGCN. However, one key limitation has not been well addressed in existing solutions: capturing long-range collaborative filtering signals, which are crucial for modeling user preference. In this work, we propose a new graph transformer (GT) framework—*Position-aware Graph Transformer for Recommendation* (PGTR), which combines the global modeling capability of Transformer blocks with the local neighborhood feature extraction of GCNs. The key insight is to explicitly incorporate node position and structure information from the user-item interaction graph into GT architecture via several purpose-designed positional encodings. The long-range collaborative signals from the Transformer block are then combined linearly with the local neighborhood features from the GCN backbone to enhance node embeddings for final recommendations. Empirical studies demonstrate the effectiveness of the proposed PGTR method when implemented on various GCN-based backbones across four real-world datasets and the robustness against interaction sparsity as well as noise. Our implementations are available in GitHub: <https://github.com/MEICRS/PGTR>.

CCS Concepts: • **Information systems** → **Recommender systems**; **Collaborative filtering**;

Additional Key Words and Phrases: Recommendation, GCN, Graph Transformer

Xiang Wang is also affiliated with Institute of Artificial Intelligence, Institute of Dataspace, Hefei Comprehensive National Science Center.

This work is supported by the National Natural Science Foundation of China (Nos. 92270114, 62302321, 62372399, and 62402470) and the Fundamental Research Funds for the Central Universities of China (Nos. WK2100000053 and PA2024GDSK0107).

Authors' Contact Information: Jiajia Chen, Institute of Dataspace, Hefei Comprehensive National Science Center, Hefei, China; e-mail: jia2chan@mail.ustc.edu.cn; Jiancan Wu (corresponding author), MoE Key Laboratory of Brain-Inspired Intelligent Perception and Cognition, University of Science and Technology of China, Hefei, China; e-mail: wujcan@gmail.com; Jiawei Chen, School of Computer Science and Technology, Zhejiang University, Hangzhou, China; e-mail: sleepyhunt@zju.edu.cn; Chongming Gao, MoE Key Laboratory of Brain-Inspired Intelligent Perception and Cognition, University of Science and Technology of China, Hefei, China; e-mail: chongming.gao@gmail.com; Yong Li, Department of Electronic Engineering, Tsinghua University, Beijing, China; e-mail: liyong07@tsinghua.edu.cn; Xiang Wang, MoE Key Laboratory of Brain-Inspired Intelligent Perception and Cognition, University of Science and Technology of China, Hefei, China; e-mail: xiangwang1223@gmail.com.



This work is licensed under Creative Commons Attribution International 4.0.

© 2025 Copyright held by the owner/author(s).

ACM 1558-2868/2025/9-ART165

<https://doi.org/10.1145/3757736>

ACM Reference format:

Jiajia Chen, Jiancan Wu, Jiawei Chen, Chongming Gao, Yong Li, and Xiang Wang. 2025. Position-aware Graph Transformer for Recommendation. *ACM Trans. Inf. Syst.* 43, 6, Article 165 (September 2025), 24 pages. <https://doi.org/10.1145/3757736>

1 Introduction

At the core of modern collaborative recommendation is learning high-quality representations of users and items from user-item interaction data. Matrix factorization [21, 35] offered a first step toward representation learning for recommendation, projecting user and item IDs into an embedding space and performing predictions based on these vectors. Subsequent work built on these foundations, augmenting the single user or item ID by incorporating additional information into the representations, such as interaction history [16, 20] and rich side information [34, 36]. More recently, approaches based on **graph convolution networks (GCNs)** have garnered significant attention in recommendation [15, 39]. It explicitly converts collaborative filtering signals into high-order connectivity patterns in the interaction graph, refining node representations through recursive neighborhood information propagation, thereby achieving **state-of-the-art (SOTA)** performance on recommendation tasks.

Despite the success, we argue that existing GCN-based methods still suffer from a core shortcoming: *inadequate capability to capture long-range collaborative filtering signals*. The signals are pivotal in recommendation systems, as they implicitly contain a wealth of potential interactions that are useful for modeling user preferences. Figure 1 shows an example of long-range collaborative signals in movie recommendation, where a less active target user might show interest in a movie seven hops away in the interaction graph. However, GCN-based methods, limited by well-recognized issues of over-smoothing [5] and over-squashing [1], can only aggregate information from few hops (e.g., $l = 2$ or 3) of neighbors [9, 15, 39]. This limitation implies that many low-degree nodes may not be adequately modeled as they have difficulty capturing long-range collaborative signals. In addition, increasing the number of layers in GCNs might lead to the propagation of noise within the interaction graph, consequently having a detrimental effect on the user experience.

In this article, we aim to answer the question “*how to capture long-range collaborative filtering signals effectively and adequately?*”. We focus on exploring of **graph transformer (GT)** technique in collaborative recommendation. The GT has been proven to effectively address the limitation of GCNs in capturing long-range information in the field of node/graph classification [33, 37, 43, 50], but it is relatively less explored in recommendation. Typically, the GT framework combines GCNs with Transformer architectures; the former could encode local neighborhood features while the latter provides long-range/global information. At the core of the GT is **positional encoding (PE)**, which is crucial for capturing graph structure [28, 29], so as to enhance the node representations. For example, Graphormer [50] devises PEs to capture the structure relation between nodes and the node importance in the graph. Compared with GCNs, the GT allows us to exploit the long-range/global information, achieving significant performance improvements in the classification tasks [28, 29].

To this end, we endeavor to introduce the advantages of the GT into recommendation. However, applying this technique to collaborative recommendation is not a trivial task, as the interaction graph data are extremely sparse, lack side information, and contain noise. Moreover, observed interactions often follow a power-law distribution [11], with significant variations in degree among different items or users. To address the aforementioned limitation of GCN-based recommenders, we propose a **position-aware graph transformer for recommendation (PGTR)** on the basis of

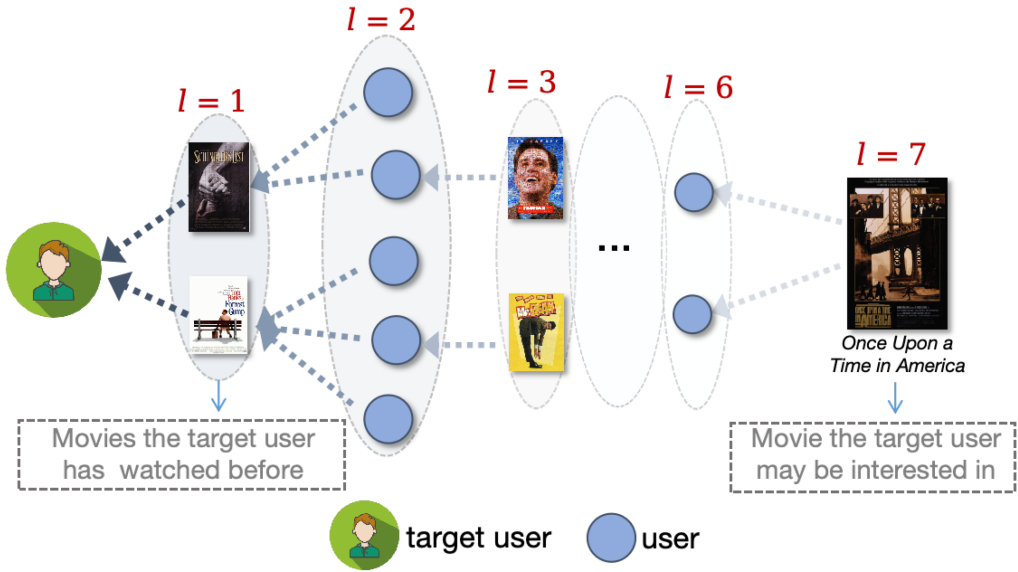


Fig. 1. An example of long-range collaborative signals in movie recommendation, where the target user may be interested in the movie *Once Upon a Time in America* seven hops away in the user-item graph. The dashed lines between nodes are the user-item interactions.

the current GCN model (e.g., LightGCN [15]). Specifically, it is composed of two key components: (1) *well-designed PEs*, which effectively encode graph structure and node positional information on the user-item interaction graph and (2) *global collaborative relations modeling*, which enables existing GCN-based models to adequately integrate long-range information for enhancing node representations. To be specific, the proposed PEs are tailored for the user-item graph in collaborative recommendation. They include *spectral encoding*, *degree encoding*, *PageRank encoding*, and *type encoding*. Among them, the *spectral encoding* encodes the positional information of the nodes, the *degree encoding* and *PageRank encoding* consider the interaction characteristics of nodes on the graph, and the *type encoding* focuses on the heterogeneity of the interaction graph. Each encoding works with a different rationality. This allows the individual information of nodes to be fully encoded on the interaction graph. After encoding position information for each node, the PEs are injected into the node representations. Subsequently, the Transformer block in PGTR models the relationships between all node pairs based on the position-aware node representations, effectuating the capture of global information. Finally, the captured global information is linearly combined with the local neighborhood features modeled by the GCN backbone, leading to the enhancement of the node representations.

It is worthwhile mentioning that our PGTR is model-agnostic. It can be applied to any GCN-based model that consists of user and item embedding. In this work, we implement it on NGCF [39], GCCF [9], LightGCN [15], and UltraGCN [27]. Experimental studies on four benchmark datasets demonstrate the effectiveness of PGTR, which significantly improves the recommendation accuracy and enhances the robustness against sparsity and noise. The contributions of this work are:

- We propose a model-agnostic GT framework, PGTR, on the basis of existing GCN-based models, which can capture long-range collaborative signals effectively and promote the performance of the GCN model.

- We highlight four node PEs tailored for the recommendation to provide interaction graph structure and node position information, which play a crucial role in our PGTR.
- We conduct extensive experiments on multiple datasets to validate the effectiveness of our PGTR. By comparing it with several SOTA methods, our approach consistently achieves improvements across various experimental settings.

2 Preliminaries

In this section, we provide an introduction to basic symbols and fundamental methods, including the GCN-based collaborative filtering method and the GT framework.

Let $\mathbf{R} \in \mathbb{R}^{N \times M}$ denote the user-item interaction matrix constructed from dataset \mathcal{D} , where N and M represent the number of users and items, respectively. \mathcal{U} and \mathcal{I} represent the user set and item set, respectively. Each entry a_{ui} in \mathbf{R} is a binary value, specifically 0 or 1, indicating whether user u has interacted with item i . We can create a user-item bipartite graph $\mathcal{B} = (\mathcal{V}, \mathcal{E})$ based on the interaction matrix \mathbf{R} . Here, \mathcal{V} consists of the set of user nodes and item nodes, while \mathcal{E} represents the set of edges. An edge exists between user u and item i if $a_{ui} = 1$. \mathcal{N}_u (\mathcal{N}_i) indicates the first-order neighbor items (users). $\text{Degree}(i)$ is the degree of node i on the graph \mathcal{B} . $\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{pmatrix}$ is the adjacency matrix.

2.1 GCN-based Collaborative Filtering

The GCN-based method is a mainstream approach in recommendation [15, 38, 39], which aggregates neighborhood information, enabling us to capture high-order connectivity patterns. The common paradigm of GCN-based models in collaborative recommendation is as follows [41]:

$$\begin{cases} \mathbf{e}_u^{(l+1)} = f_{\text{combine}}(\mathbf{e}_u^{(l)}, f_{\text{aggregate}}(\{\mathbf{e}_i^{(l)} | i \in \mathcal{N}_u\})), \\ \mathbf{e}_i^{(l+1)} = f_{\text{combine}}(\mathbf{e}_i^{(l)}, f_{\text{aggregate}}(\{\mathbf{e}_u^{(l)} | u \in \mathcal{N}_i\})). \end{cases} \quad (1)$$

For a user node u , by aggregating the representations of its neighbors $\{\mathbf{e}_i^{(l)} | i \in \mathcal{N}_u\}$ and incorporating its own representation $\mathbf{e}_u^{(l)}$ at the l th layer, the updated representation $\mathbf{e}_u^{(l+1)}$ is obtained. This holds true for each item node as well. After L rounds of aggregation, the GCN model encodes the L -order neighbors. Subsequently, we obtain node representations for L layers, resulting in the final node representation through a readout function:

$$\mathbf{e} = f_{\text{readout}}(\{\mathbf{e}^{(l)} | l = 0, \dots, L\}), \quad (2)$$

where $\mathbf{e}^{(0)} \in \mathbb{R}^d$ is the initialized embeddings, d is the embedding size. To be specific, there are a number of designs for $f_{\text{combine}}(\cdot)$, $f_{\text{aggregate}}(\cdot)$, and $f_{\text{readout}}(\cdot)$. More details can be referred to the survey [44].

Traditional GCN-based methods can only capture collaborative signals through message passing and neighbor aggregation. However, when it comes to capturing long-range collaborative signals, such as $L > 3$, GCN-based models often face the issues of over-smoothing and over-squashing [1, 5]. This may result in a performance decline and the long-range collaborative signals cannot be captured. The GT framework has the potential to address these issues. Next, we introduce the GT framework.

2.2 GT

The GT techniques have been applied to various tasks of graph learning, such as node classification [6, 43, 54] and graph classification [18, 30]. Typically, the GT framework combines GCN module with the Transformer module to leverage their respective strengths. The GCN module is utilized

for learning the local structure of the graph, while the Transformer provides global information. By combining these two components in various configurations, the GT can effectively model both local and global relationships, resulting in improved performance for specific tasks [28].

As mentioned before, the GCN model has been introduced. In this part, we focus on introducing the vanilla Transformer architecture. Assuming that the input of node i to the l th Transformer layer is denoted as $\mathbf{z}_i^{(l)}$, we compute attention scores within the Transformer across the full-graph (including T nodes) to estimate the latent relationships between node i and other nodes. For simplicity, there is

$$\tilde{a}_{ij}^{(l)} = \frac{\exp\left(\left(\mathbf{W}_Q^{(l)} \mathbf{z}_i^{(l)}\right)^\top \left(\mathbf{W}_K^{(l)} \mathbf{z}_j^{(l)}\right)\right)}{\sum_{t=1}^T \exp\left(\left(\mathbf{W}_Q^{(l)} \mathbf{z}_i^{(l)}\right)^\top \left(\mathbf{W}_K^{(l)} \mathbf{z}_t^{(l)}\right)\right)}, \quad (3)$$

then perform global aggregation of node information:

$$\mathbf{z}_i^{(l+1)} = \sum_{j=1}^T \tilde{a}_{ij}^{(l)} \cdot \left(\mathbf{W}_V^{(l)} \mathbf{z}_j^{(l)}\right), \quad (4)$$

where $\mathbf{W}_Q^{(l)}$, $\mathbf{W}_K^{(l)}$, and $\mathbf{W}_V^{(l)}$ are learnable parameters at the l th Transformer layer. The updating for all T nodes in one layer using Equations (3) and (4) requires prohibitive $\mathcal{O}(T^2)$ complexity [43]. When dealing with a large graph (with up to 2M nodes), computation becomes challenging. In order to accelerate the computation, Nodeformer [43] proposes an efficient approximate algorithm named kernelized message passing method (a.k.a. Nodeformer Convolution) that reduces the computational complexity from $\mathcal{O}(T^2)$ to $\mathcal{O}(T)$. In brief, Equations (3) and (4) are finally transformed into

$$\left(\mathbf{z}_i^{(l+1)}\right)^\top \approx \frac{\phi\left(\mathbf{W}_Q^{(l)} \mathbf{z}_i^{(l)}\right)^\top \sum_{j=1}^T \phi\left(\mathbf{W}_K^{(l)} \mathbf{z}_j^{(l)}\right) \cdot \left(\mathbf{W}_V^{(l)} \mathbf{z}_j^{(l)}\right)^\top}{\phi\left(\mathbf{W}_Q^{(l)} \mathbf{z}_i^{(l)}\right) \cdot \sum_{t=1}^T \phi\left(\mathbf{W}_K^{(l)} \mathbf{z}_t^{(l)}\right)^\top}, \quad (5)$$

where $\phi(\cdot) : \mathbb{R}^h \rightarrow \mathbb{R}^m$ is a low-dimensional feature map with random transformation [43]. For example,

$$\phi(\mathbf{x}) = \frac{\exp\left(\frac{-\|\mathbf{x}\|_2^2}{2}\right)}{\sqrt{m}} \left[\exp\left(\mathbf{w}_1^\top \mathbf{x}\right), \dots, \exp\left(\mathbf{w}_m^\top \mathbf{x}\right)\right], \quad (6)$$

where $\mathbf{x} \in \mathbb{R}^h$ is the input, $\mathbf{w}_k \in \mathbb{R}^h \sim \mathcal{N}(0, \mathbf{I}_h)$ (\mathbf{I}_h is the identity matrix) is an i.i.d. sampled random transformation [10]. This makes global node aggregation on large-scale graphs feasible.

3 Methodology

In this section, we present our proposed PGTR framework, which incorporates GCN-based backbone (e.g., LightGCN [15], NGCF [39]), Transformer module (here we use the Nodeformer Convolution [43] introduced in Section 2.2) and well-designed node PEs. The GCN-based backbone encodes local neighborhood features, the Transformer module is used for capturing global collaborative relations, as well as PEs are utilized to provide structure and position information of nodes on the interaction graph within the Transformer block.

We start by introducing four types of node PEs tailored for collaborative recommendation. Then, we give the detailed implementations for our PGTR framework combined with the GCN-based backbone and the Transformer module.

3.1 Proposed PEs

In practice, PEs are widely used in the GT for improving the capability of modeling the relations among nodes [2, 12, 22, 53]. They provide the graph structure and position information of nodes within the Transformer module. However, they are not tailored for the field of collaborative recommendation. For example, it is crucial to not only consider the relationship between users and items but also take into account the relationship among users (user-user) and items (item-item). In addition, there are two types of nodes (user/item nodes) that need to be taken into consideration. In the next, we will introduce each of the PEs proposed for collaborative recommendation.

3.1.1 Spectral Encoding. In graph learning tasks, Laplacian PE [12, 22, 32, 33] is commonly utilized in the GT from the spectral domain of graph. This PE can encode relative position information between nodes, and neighboring nodes would have similar encodings. In this article, we adopt this method for encoding the position information of each user and item in the user-item interaction graph. Formally, for the interaction adjacency matrix \mathbf{A} , we precomputed its Laplacian eigenvectors,

$$\mathbf{S}^\top \mathbf{A} \mathbf{S} = \mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}, \quad (7)$$

where \mathbf{D} is the degree matrix of \mathbf{A} , \mathbf{I} is the identity matrix, $\mathbf{\Lambda}$ and \mathbf{S} are the eigenvalues and eigenvectors, respectively. Generally, eigenvectors of the H_C smallest non-trivial eigenvalues are used as the Laplacian PEs. Thus, we obtain the Laplacian PEs $\mathbf{P}^{C_0} = [\mathbf{U}_0, \mathbf{V}_0] \in \mathbb{R}^{H_C \times (N+M)}$ from \mathbf{S} , where $\mathbf{U}_0 \in \mathbb{R}^{H_C \times N}$ and $\mathbf{V}_0 \in \mathbb{R}^{H_C \times M}$ represent the Laplacian PEs of users and items, respectively.

In recommendation, building user-user and item-item relationships is of great significance [14]. Therefore, we encode position information for each user and item on the user-side and item-side interaction graph, respectively. First, we construct the user-side adjacency matrix $\mathbf{A}_U \in \mathbb{R}^{N \times N}$ and the item-side adjacency matrix $\mathbf{A}_I \in \mathbb{R}^{M \times M}$, where the elements are 0 or 1, indicating the existence of second-order connectivity in \mathcal{B} . Then, we have

$$\begin{cases} \mathbf{S}_1^\top \mathbf{\Lambda}_1 \mathbf{S}_1 = \mathbf{L}_U = \mathbf{I}_U - \mathbf{D}_U^{-1/2} \mathbf{A}_U \mathbf{D}_U^{-1/2}, \\ \mathbf{S}_2^\top \mathbf{\Lambda}_2 \mathbf{S}_2 = \mathbf{L}_I = \mathbf{I}_I - \mathbf{D}_I^{-1/2} \mathbf{A}_I \mathbf{D}_I^{-1/2}. \end{cases} \quad (8)$$

Likewise, we select the eigenvectors corresponding to the H_C smallest non-trivial eigenvalues as the Laplacian PEs for users and items on their respective interaction graphs (user-side and item-side). That is, we obtain $\mathbf{U}_1 \in \mathbb{R}^{H_C \times N}$ from \mathbf{S}_1 , and $\mathbf{V}_1 \in \mathbb{R}^{H_C \times M}$ from \mathbf{S}_2 . Further, there is $\mathbf{P}^{C_1} = [\mathbf{U}_1, \mathbf{V}_1] \in \mathbb{R}^{H_C \times (N+M)}$.

In the end, we combine \mathbf{P}^{C_0} and \mathbf{P}^{C_1} to create *Spectral Encodings*, i.e.,

$$\mathbf{P}^C = (1 - \lambda_C) \mathbf{P}^{C_0} + \lambda_C \mathbf{P}^{C_1}, \quad (9)$$

where $\lambda_C \in [0, 1]$ is the hyper-parameter to control the proportions of different Laplacian PEs. Each user (item) can be determined by their spectral encodings separately on the user-item interaction graph and the user-side interaction graph (item-side interaction graph). This helps the model to have a clearer understanding of the relationships between users and items.

3.1.2 Degree Encoding. In recommendation systems, the popularity of items and the activity of users are important indicators that reflect the quality of items and user behavior patterns. From the perspective of items, when two items have similar popularity, they tend to appear in similar contexts or scenarios, such as in similar recommendation lists, purchase histories, or social circles. For example, some users are consistently drawn to popular songs, while others focus on niche or lesser-known tracks. Therefore, items with similar popularity are more likely to exhibit connections in user behavior data. From the perspective of users, when two users have similar activity, it implies

that they exhibit similar behavior patterns and engagement on the platform. For example, based on similar preferences, they may collectively discover and explore similar items, or they may mutually follow each other on social networks. To this end, we devise degree encodings for each user and item.

The degree distribution in recommendation datasets is skewed. For instance, in the Amazon-book dataset [15], some items have thousands of interactions, while others only have a dozen. If we encode each degree as done in Graphormer [50], it becomes challenging to establish connections between nodes with close degrees. Therefore, we adopt a strategy of grouping degrees, as shown in Figure 3.

Specifically, for the item set, each item $i \in \mathcal{I}$ has an associated popularity $p_i = \text{Degree}(i)$, i.e.,

$$S_I = \{(i_1, p_{i_1}), (i_2, p_{i_2}), \dots, (i_M, p_{i_M})\}. \quad (10)$$

We sort the items in ascending order of their popularity and uniformly divide these items into N_d groups. For an item i , the degree PE is

$$\mathbf{P}_i^D = DE_I(N_d^i), 0 \leq N_d^i < N_d, \quad (11)$$

where $DE_I(\cdot)$ is the degree group encoder for the items, N_d^i is the group that the item i belongs to. Finally, we obtain the degree encoding for all items, denoted as $\mathbf{P}_I^D \in \mathbb{R}^{H_D \times M}$, H_D is the encoding dimension.

Similarly, each user $u \in \mathcal{U}$ has a corresponding activity $a_u = \text{Degree}(u)$ and $S_U = \{(u_1, a_{u_1}), (u_2, a_{u_2}), \dots, (u_N, a_{u_N})\}$. We sort the users in ascending order of their activity and also uniformly divide them into N_d groups. The degree PE of user u is

$$\mathbf{P}_u^D = DE_U(N_d^u), 0 \leq N_d^u < N_d, \quad (12)$$

where $DE_U(\cdot)$ is the degree group encoder for the users, N_d^u is the group that the user u belongs to. Further, we can get $\mathbf{P}_U^D \in \mathbb{R}^{H_D \times N}$ for all users.

It is necessary to note that this encoding is not entirely valid for items (or users) in the critical position within a group because there are some items (or users) with similar popularity (activity) located in different groups. But in general, we make this PE work by adjusting N_d .

3.1.3 PageRank Encoding. PageRank [31] can be used to describe the importance or influence of items and users on the interaction graph. The PageRank value of an item (or user) is not only related to its own degree but also depends on the importance of its neighbors on the graph. When two items have similar PageRank values, it indicates that they are similarly favored by users. For users, when two users have similar PageRank values, their interaction patterns tend to be similar. Therefore, we propose PageRank encodings, as shown in Figure 4.

First, we precompute PageRank values for all users and items based on the interaction graph \mathcal{B} .¹ Then, each item i has a PageRank value r_i and there is

$$Q_I = \{(i_1, r_{i_1}), (i_2, r_{i_2}), \dots, (i_M, r_{i_M})\}. \quad (13)$$

We sort the items in ascending order of their PageRank values and divide these items into N_r groups evenly. For an item i , its PageRank PE is

$$\mathbf{P}_i^R = PE_I(N_r^i), 0 \leq N_r^i < N_r, \quad (14)$$

where $PE_I(\cdot)$ is the PageRank group encoder for the items, N_r^i is the group that the item i belongs to. The PageRank PEs $\mathbf{P}_I^R \in \mathbb{R}^{H_R \times M}$ for all items can be obtained, H_R is the encoding dimension. Similar to Section 3.1.2, from the user's perspective, we obtain the PageRank PEs corresponding to all users $\mathbf{P}_U^R \in \mathbb{R}^{H_R \times N}$.

¹We utilize `networkx.pagerank()` in Python to compute PageRank values.

Implementing this encoding through grouping faces similar challenges as the *Degree Similarity* criterion. However, by adjusting the number of groups, we also achieve its effectiveness in constructing the relationships among users and items. For simplicity, we set the number of groups for both items and users as N_r .

3.1.4 Type Encoding. Users and items have different relation patterns and attributes in a recommendation system. For example, the display of items on a Web site or app (such as organized sorting by category or popularity) is different from the relationships between users. By distinguishing the node types on the interaction graph, the PGTR can better understand the interaction and connection among users and items. To this end, we propose type encodings based on the node types.

Intuitively, we assign a type of PE to item groups and user groups separately. All items share the same type of PE, and all users share the same one as well, i.e.,

$$\begin{cases} \mathbf{P}_i^Y = TE(0), i \in \mathcal{I}, \\ \mathbf{P}_u^Y = TE(1), u \in \mathcal{U}. \end{cases} \quad (15)$$

where $TE(\cdot)$ is the node type encoder. We can get $\mathbf{P}_I^Y \in \mathbb{R}^{H_Y \times M}$ and $\mathbf{P}_U^Y \in \mathbb{R}^{H_Y \times N}$ for all items and users respectively, H_Y is the encoding dimension.

Overall, the proposed four PEs provide a thoughtful consideration of the relationship among users and items from different perspectives, enabling the capture and modeling of various relationships. Next, we introduce the PGTR framework and incorporate the four PEs into the framework to facilitate the learning of the relationships.

3.2 PGTR Framework

In this section, we integrate Equation (5) (i.e., Nodeformer Convolution) and several proposed PEs into the existing GCN-based model (e.g., LightGCN [15], NGCF [39]) to form a new PGTR model. Figure 2 illustrates the overall framework. Specifically, we first inject positional information into the initialization of node embeddings in the model, enabling position identification at the node representation. For any given node j (user or item), there is

$$\mathbf{h}_j^{(0)} = \mathbf{e}_j^{(0)} + \lambda_1 \mathbf{P}_j, \quad (16)$$

where $\lambda_1 \in [0, 1]$ controls the strength of the positional information, and

$$\mathbf{P}_j = \begin{cases} \mathbf{W}_I(\mathbf{W}_C \mathbf{P}_j^C + \mathbf{W}_D \mathbf{P}_j^D + \mathbf{W}_R \mathbf{P}_j^R + \mathbf{W}_Y \mathbf{P}_j^Y), & \text{if } j \in \mathcal{I}, \\ \mathbf{W}_U(\mathbf{W}_C \mathbf{P}_j^C + \mathbf{W}_D \mathbf{P}_j^D + \mathbf{W}_R \mathbf{P}_j^R + \mathbf{W}_Y \mathbf{P}_j^Y), & \text{if } j \in \mathcal{U}, \end{cases} \quad (17)$$

where $\mathbf{W}_I \in \mathbb{R}^{d \times d}$, $\mathbf{W}_U \in \mathbb{R}^{d \times d}$, $\mathbf{W}_C \in \mathbb{R}^{d \times H_C}$, $\mathbf{W}_D \in \mathbb{R}^{d \times H_D}$, $\mathbf{W}_R \in \mathbb{R}^{d \times H_R}$, and $\mathbf{W}_Y \in \mathbb{R}^{d \times H_Y}$ are learnable transformation matrices that transform the PE dimension to the embedding dimension. Next, $\mathbf{h}_j^{(0)}$ is fed to the GCN module and Nodeformer Convolution module. At the l th Transformer layer, we have the following steps:

$$\tilde{\mathbf{h}}_j^{(l+1)} = f_{\text{combine}}(\mathbf{h}_j^{(l)}, f_{\text{aggregate}}(\{\mathbf{h}_k^{(l)} | k \in \mathcal{N}_j\})), \quad (18)$$

this step models local neighborhood features through the GCN module. Thereafter, for capturing global collaborative signals, based on Equation (5),

$$\left(\tilde{\mathbf{h}}_j^{(l+1)}\right)^\top \approx \frac{\phi\left(\mathbf{W}_Q^{(l)} \hat{\mathbf{h}}_j^{(l)}\right)^\top \sum_{v=1}^{N+M} \phi\left(\mathbf{W}_K^{(l)} \hat{\mathbf{h}}_v^{(l)}\right) \cdot \left(\mathbf{W}_V^{(l)} \hat{\mathbf{h}}_v^{(l)}\right)^\top}{\phi\left(\mathbf{W}_Q^{(l)} \hat{\mathbf{h}}_j^{(l)}\right)^\top \sum_{t=1}^{N+M} \phi\left(\mathbf{W}_K^{(l)} \hat{\mathbf{h}}_t^{(l)}\right)}, \quad (19)$$

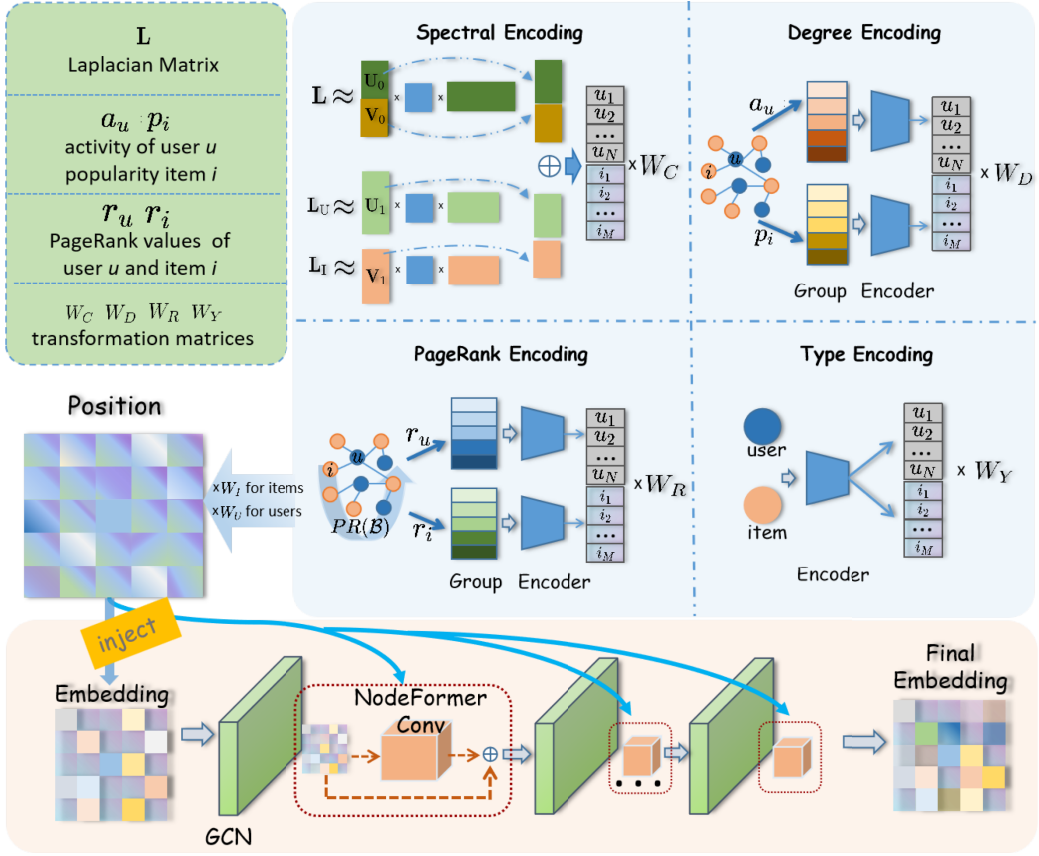


Fig. 2. Overall framework of our PGTR model.

where

$$\hat{\mathbf{h}}_j^{(l)} = \tilde{\mathbf{h}}_j^{(l+1)} + \lambda_2 \mathbf{P}_j, \quad (20)$$

which means that the positional information is re-injected into the output representation of the GCN module, $\lambda_2 \in [0, 1]$ controls the injection strength as well. Finally, we get the representation of node j at this layer,

$$\mathbf{h}_j^{(l+1)} = (1 - \lambda_3) \tilde{\mathbf{h}}_j^{(l+1)} + \lambda_3 \bar{\mathbf{h}}_j^{(l+1)}, \quad (21)$$

where $\lambda_3 \in [0, 1]$ controls the ratio of mixing between local and global information. After L layers, there is the final node representation derived by our PGTR framework, i.e.,

$$\mathbf{h}_j = f_{readout}(\{\mathbf{h}_j^{(l)} | l = 0, \dots, L\}). \quad (22)$$

Note that in our implementations, we omit $\mathbf{W}_Q^{(l)}$, $\mathbf{W}_K^{(l)}$, and $\mathbf{W}_V^{(l)}$ in the Transformer for simplicity.

Training Loss: We train our PGTR model using the **sampled softmax (SSM)** loss [42], i.e.,

$$\mathcal{L}_{SSM} = -\frac{1}{|\mathcal{D}|} \sum_{(u,i) \in \mathcal{D}} \log \frac{\exp(f(u,i))}{\exp(f(u,i)) + \sum_{j \in \mathcal{N}} \exp(f(u,j))}, \quad (23)$$

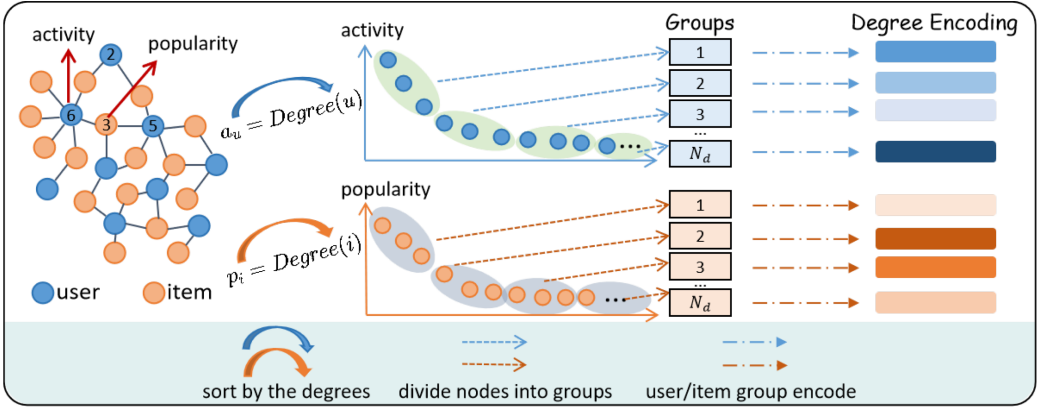


Fig. 3. The procedure of degree encoding for users and items.

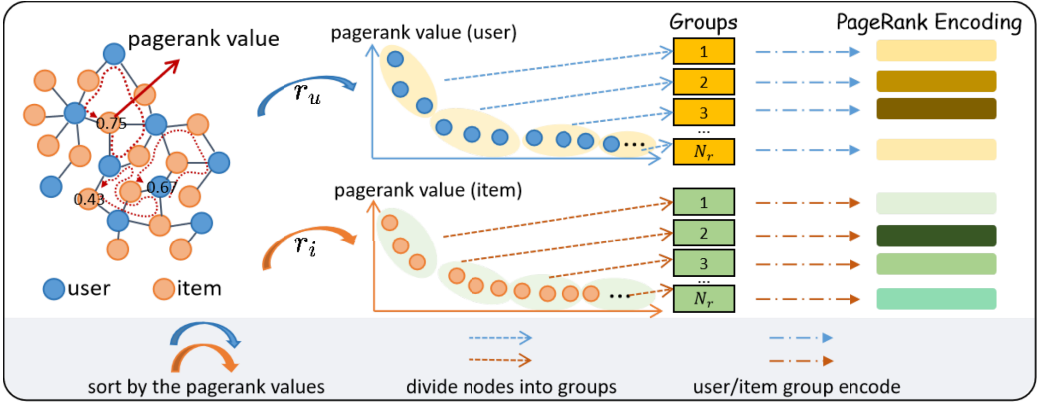


Fig. 4. The procedure of PageRank encoding for users and items.

where $f(u, i) = \frac{s_u^T s_i}{\tau}$, s_u and s_i are the normalized representations of \mathbf{h}_u and \mathbf{h}_i , respectively, τ is the temperature coefficient, $j \in \mathcal{N}$ is the negative sample of user u . We adopt the in-batch negative trick [17] just like [42] for parallel computing, which treats positive items of other users in the same batch as the negatives. The training procedure for our proposed method is outlined in Algorithm 1.

Complexity Analysis: Note that among the four PEs, \mathbf{P}^D , \mathbf{P}^R , and \mathbf{P}^Y are trainable. Considering the space complexity of trainable parameters, the PGTR only adds a small number of trainable parameters compared to the original GCN-based model, i.e.,

$$O(2(N_d H_D + N_r H_R + H_Y) + d(H_C + H_D + H_R + H_Y + 2d)).$$

It is much smaller than all embedding parameters $O((N + M)d)$ for large N and M .

Additionally, we provide a brief analysis of the theoretical time complexity for the proposed PGTR model and compared methods. Table 1 summarizes the time complexity and key components of each method. The PGTR methods, which incorporate the Transformer architecture, exhibit higher complexities due to the additional computational requirements of the Transformer module.

Based on the observed experiments, for the two small datasets (Amazon-elec and Douban-book), the single-round training time of PGTR is similar to that of LightGCN. However, since PGTR establishes relationships between nodes extensively during training, it achieves faster performance

Table 1. Theoretical Time Complexity of Compared Methods

Method	Time Complexity	Key Components
BPR-MF	$O(\mathcal{E} \cdot d)$	Matrix factorization, pairwise loss
NGCF	$O(L \cdot (\mathcal{E} \cdot d + T \cdot d^2))$	GCN layers (L), neighborhood aggregation
LightGCN	$O(L \cdot \mathcal{E} \cdot d)$	Simplified GCN (no nonlinear operations)
UltraGCN	$O(\mathcal{E} \cdot d)$	High-order proximity via loss regularization
SGL	$O(L \cdot \mathcal{E} \cdot d + \mathcal{E} _{\text{aug}} \cdot d)$	GCN + contrastive learning
SimGCL	$O(L \cdot \mathcal{E} \cdot d + T \cdot d)$	GCN + noise augmentation
LightGCL	$O(L \cdot \mathcal{E} \cdot d + \text{SVD})$	GCN + SVD-based augmentation
GFormer	$O(L \cdot (T \cdot d^2 + \mathcal{E} \cdot d))$	LightGCN backbone + Collaborative Rationale Discovery
LightGCN-PGTR	$O(L \cdot (T^2 \cdot d^2 + \mathcal{E} \cdot d))$	LightGCN backbone + Linearized Transformer
UltraGCN-PGTR	$O(T^2 \cdot d^2 + \mathcal{E} \cdot d)$	UltraGCN backbone + Linearized Transformer

Algorithm 1: Training Algorithm of PGTR

```

1: Data: User-item interaction history  $\mathcal{D}$  between  $\mathcal{U}$  and  $\mathcal{I}$ ;
2: Input: Batch size  $B$ , embedding size  $d$ , number of GCN layers  $L$ , Spectral encoding dimension  $H_C$ , Groups  $N_d$ ,
   Degree encoding dimension  $H_D$ , Groups  $N_r$ , PageRank encoding dimension  $H_R$ , Type encoding dimension  $H_Y$ ,  $\{\lambda_C, \lambda_1, \lambda_2, \lambda_3\}$ ;
3: Output: Trained model parameters  $\Theta = \{\mathbf{E}^{(0)}, \mathbf{W}_Q^{(l)}, \mathbf{W}_K^{(l)}, \mathbf{W}_V^{(l)}, \mathbf{P}^D, \mathbf{P}^R, \mathbf{P}^Y, \mathbf{W}_I, \mathbf{W}_U, \mathbf{W}_C, \mathbf{W}_D, \mathbf{W}_R, \mathbf{W}_Y\}$ .
4: Initialize user and item embeddings  $\mathbf{E}^{(0)}$  (i.e.,  $\mathbf{e}_u^{(0)}, \mathbf{e}_i^{(0)} \in \mathbb{R}^d$ );
5: Construct user-item interaction matrix  $\mathbf{R}$  from the dataset  $\mathcal{D}$ ;
6: Compute adjacency matrix  $\mathbf{A}$  and degree matrix  $\mathbf{D}$  from  $\mathbf{R}$ ;
7: Compute Spectral Encodings  $\mathbf{P}^C$  (untrainable) for all nodes; ▷ Section 3.1.1
8: Compute Degree Encodings  $\mathbf{P}^D$  (trainable) for all nodes; ▷ Section 3.1.2
9: Compute PageRank Encodings  $\mathbf{P}^R$  (trainable) for all nodes; ▷ Section 3.1.3
10: Compute Type Encodings  $\mathbf{P}^Y$  (trainable) for all nodes; ▷ Section 3.1.4
11: Initialize positional encodings  $\mathbf{P}$  for all user and item nodes; ▷ Eq. (17)
12: while not done do
13:   for each batch  $b \in \mathcal{D}$  do
14:     Inject positional information  $\mathbf{P}$  into the initialization of node embeddings  $\mathbf{E}^{(0)}$ ; ▷ Eq. (16)
15:     for  $l = 1$  to  $L$  do
16:       Model local neighborhood features through the GCN module; ▷ Eq. (18)
17:       Re-inject the positional information into the node representations; ▷ Eq. (20)
18:       Capture global collaborative signals through the Nodeformer Convolution module; ▷ Eq. (19)
19:       Mix local and global information and get the representation of nodes at this layer; ▷ Eq. (21)
20:     end for
21:     Get the final node representations; ▷ Eq. (22)
22:     Calculate the sampled softmax loss  $\mathcal{L}_{SSM}$  in the batch  $b$ ; ▷ Eq. (23)
23:     Update model parameters  $\Theta$  using gradient descent on the loss  $\mathcal{L}_{SSM}$ .
24:   end for
25: end while

```

improvement and requires fewer training rounds. On the larger LastFM and Amazon-book datasets, the single-round training time of PGTR is approximately $5\times$ that of LightGCN. However, the overall training duration is only $1.2\times$ longer on the LastFM dataset and $2.9\times$ longer on the Amazon-book dataset. The comparison of computation efficiency between PGTR and the base model is shown in Section 4.6.

4 Experiments

In this section, we conduct experiments to evaluate the performance of our proposed PGTR. Our experiments are expected to answer the following **research questions (RQs)**:

- *RQ1*: How effective is the proposed PGTR compared to other SOTA models?
- *RQ2*: How well does the proposed PGTR perform in long-range recommendations?
- *RQ3*: How does the proposed method perform on different levels of sparsity and noisy data?
- *RQ4*: Are the four proposed PEs effective?

4.1 Experimental Setup

We perform experiments on four commonly used real-world datasets to evaluate our method: Amazon-electronics, Douban-book, LastFM, and Amazon-book. Table 3 provides an overview of the statistics for these datasets. In order to simulate different levels of sparsity, we divide the dataset into 20%/40%/60%/80% as the training set. Within the training set, 20% is used as a validation set to tune hyper-parameters. The remaining 80%/60%/40%/20% is treated as the testing set. Two widely used evaluation metrics are adopted, i.e., Recall@20 and NDCG@20.

4.1.1 Compared Methods. To validate the superiority of our method, we compare it with several advanced baselines used for collaborative filtering, including several well-known Matrix factorization based and GCN-based methods: *BPRMF* [21], *NGCF* [39], *GCCF* [9], *LightGCN* [15], and *UltraGCN* [27], and several SOTA self-supervised learning-based methods:

- *SGL* [41]: It employs various random data augmentation techniques, such as edge dropping, node dropping, and random walks, to generate diverse views of interaction structures. These views are then used for contrastive learning.
- *EGLN* [49]: It incorporates two modules: a node embedding learning module and a graph structure learning module. They are designed for learning from each other and enhancing the representations.
- *SimGCL* [51]: It enhances LightGCN by incorporating a noise-based augmentation at the user/item representation level and contrastive learning.
- *LightGCL* [3]: It introduces a graph augmentation strategy based on singular value decomposition to extract global collaborative patterns effectively. Contrastive learning is conducted between the reconstructed graph and the original graph.
- *GFormer* [23]: This model enables the construction of rationale-aware general augmentation through masked graph autoencoding.

4.1.2 Hyper-parameter Settings. For a fair comparison, the GCN-based method that requires specifying the number of GCN layers is set to two layers. The embedding size for all methods is set to 32. All models use the Adam optimizer [19]. Except for the Amazon-elec dataset, which has a batch size of 64, the batch sizes for the other three datasets are set to 2,048. For our PGTR method, H_C is searched from $\{1, 10, 50, 100, 150, 200, 250, 300\}$, with a default value of 50. H_D , H_R , and H_Y are searched from $\{2, 4, 8, 16, 32, 64\}$, with a default value of 4. N_d and N_r are searched from $\{1, 5, 10, 15, 20\}$, with a default value of 10. For simplicity, λ_C , λ_1 , and λ_2 are all searched in $\{0, 1\}$. And the temperature τ of the SSM loss is searched in a coarse grain range of $\{0.02, 0.1, 0.2, \dots, 1.0, 1.2\}$.

4.2 Performance Comparison (RQ1)

In Table 2, we present a performance comparison between our proposed method and other methods, where the dataset splitting is training:testing = 2:8. Specifically, we apply the PGTR method to four existing GCN-based recommender models, enabling each model to capture long-range/global collaborative signals. Overall, our proposed method achieves improved performance over the four GCN-based models and attains the best performance on various datasets in such a sparse scene. Regarding Table 2, the following observations can be made. (1) Overall, the GCN-based method outperforms the MF method, which is consistent with the current mainstream conclusion.

Table 2. Performance Comparison between Our Method PGTR and Other Counterparts

Data	Amazon-Elec		Douban-Book		LastFM		Amazon-Book	
Model	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF [2009]	0.0296	0.0410	0.0487	0.0837	0.0478	0.1054	0.0191	0.0361
NGCF [2019]	0.0684	0.0921	0.0735	0.1164	0.0263	0.0654	0.0141	0.0286
GCCF [2020]	0.0352	0.0534	0.0570	0.1114	0.0325	0.0840	0.0179	0.0364
LightGCN [2020]	0.0330	0.0485	0.0677	0.1278	0.0535	0.1215	0.0250	0.0472
UltraGCN [2021]	<u>0.0781</u>	0.0955	<u>0.1123</u>	<u>0.1966</u>	<u>0.0612</u>	<u>0.1361</u>	0.0212	0.0398
EGLN [2021]	0.0551	0.0732	0.0726	0.1353	0.0543	0.1227	0.0238	0.0454
SGL [2021]	0.0355	0.0465	0.0804	0.1490	0.0551	0.1235	<u>0.0260</u>	0.0463
SimGCL [2022]	0.0381	0.0510	0.0809	0.1517	0.0540	0.1221	0.0245	0.0432
LightGCL [2023]	0.0432	0.0610	0.0682	0.0904	0.0570	0.1217	0.0250	0.0428
GFormer [2023]	0.0761	<u>0.1055</u>	0.0781	0.1468	0.0577	0.1238	0.0258	<u>0.0489</u>
NGCF-PGTR	0.0802	0.1084	0.0909	0.1558	0.0616	0.1394	0.0274	0.0506
GCCF-PGTR	0.0796	0.1081	0.0889	0.1584	0.0579	0.1258	0.0264	0.0476
LightGCN-PGTR	0.0811	0.1103	0.0899	0.1536	0.0610	0.1360	0.0290	0.0521
UltraGCN-PGTR	0.0809	0.1087	0.1434	0.2287	0.0629	0.1395	0.0216	0.0404
Improve	3.84%	4.55%	27.69%	16.33%	2.78%	2.50%	11.54%	6.54%

The best result is bolded and the runner-up is underlined. “[2020]” indicates the publication year of the corresponding method. Recall and NDCG scores are reported at a cutoff of 20.

Table 3. Dataset Description

Dataset	#Users	#Items	#Interactions	Density
Amazon-elec	1,435	1,522	35,931	0.062%
Douban-book	13,024	22,347	792,062	0.272%
LastFM	23,566	48,123	3,034,763	0.268%
Amazon-book	52,643	91,599	2,984,108	0.062%

This indicates the importance of utilizing neighbor information and capturing local collaborative relations. (2) Methods based on self-supervised learning demonstrate strong recommendation performance. However, we also observe that in cases where the training set is very sparse, some self-supervised learning methods perform similarly to the GCN-based models, such as SimGCL on the Amazon-book dataset, where its performance is lower than the base model LightGCN. This may be attributed to over-fitting, suggesting that relying solely on local information is insufficient for sparse data. (3) Our proposed method achieves significant performance improvements across various backbones. On the one hand, by incorporating the position information of nodes into the model, our method can better capture relations among nodes. On the other hand, employing the Transformer architecture allows the model to capture long-range/global collaborative signals, compensating for the limitations of the existing GCN-based models.

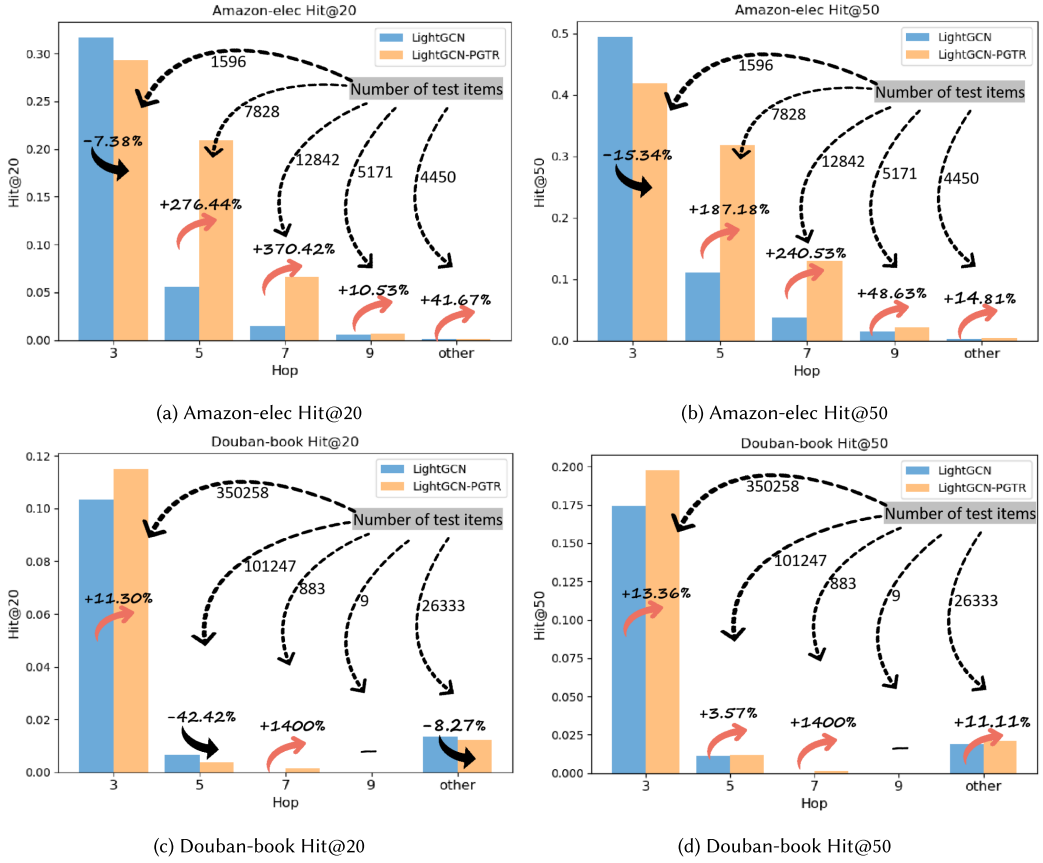


Fig. 5. Performance comparison of LightGCN and LightGCN-PGTR for different hop distances on the Amazon-elec and Douban-book datasets, respectively. “Number of test items” refers to the number of test items that appear in the user’s interaction nodes at each hop in the training interaction graph. For example, in (a), the value 1,596 at Hop = 3 indicates that there are 1,596 interacted items in the Amazon-elec test set that appear in the three-hop neighbors of users in the training interaction graph. The metric Hit@K represents the proportion of correctly recommended items within the Top-K recommendations for each user, corresponding to the true items at each hop.

4.3 Performance Comparison on Long-range Item Recommendation (RQ2)

In this section, we compare the performance of LightGCN and LightGCN-PGTR on long-range item recommendation across two datasets: Amazon-elec and Douban-book. The goal is to assess how effectively each model can recommend items that are multiple hops away from users in the interaction graph. The results are shown in Figures 5 and 6.

4.3.1 Overall Performance Comparison. In Figure 5, the “Number of test items” indicates the number of test items that appear in the user’s interaction nodes at each hop in the training interaction graph. For instance, at Hop = 3, the value 1,596 indicates that there are 1,596 interacted items in the Amazon-elec test set that appear in the three-hop neighbors of users in the training interaction graph. The “other” category refers to items that are either further away or not connected through any path. The metric used is Hit@K, which measures the proportion of correctly recommended items within the Top-K recommendations for each user, corresponding to the true items at each hop.

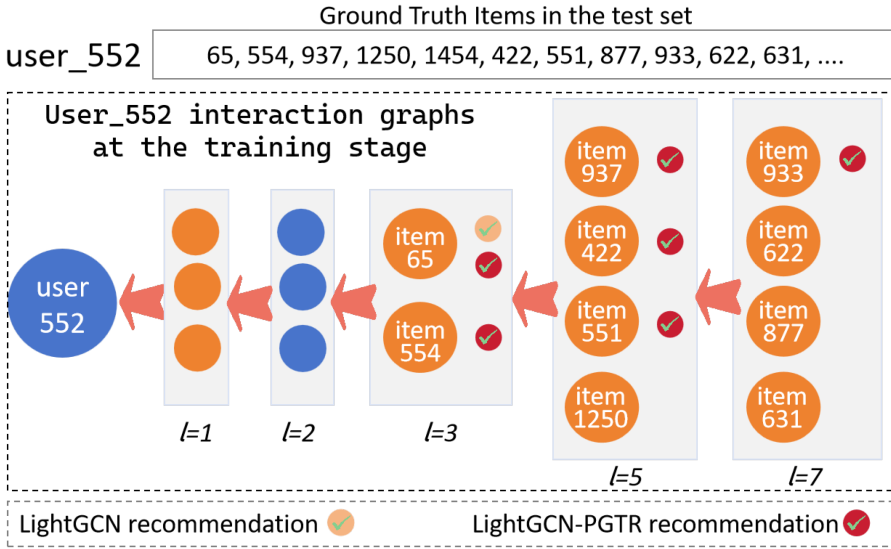


Fig. 6. Case study illustrating the long-range recommendation results of LightGCN and LightGCN-PGTR on the Amazon-elec. We omit the detailed interactions of user-552 for clarity. We use l to denote the reachable nodes of user-552 at the l th hop. Additionally, to enhance the readability of the figure, we exclude the reachable user nodes at $l = 4$ and $l = 6$.

Figure 5(a) and (b) shows the performance of LightGCN and LightGCN-PGTR on the Amazon-elec dataset for different hop distances. LightGCN demonstrates superior performance in recommending items that are closely connected to users (three hops). This suggests that LightGCN is effective in capturing local collaborative signals. For items that are five or more hops away, LightGCN-PGTR significantly outperforms LightGCN. This indicates that PGTR enhances the model's ability to capture long-range collaborative signals, which are crucial for modeling user preferences accurately. Figure 5(c) and (d) presents the performance on the Douban-book dataset. Similar to the Amazon-elec dataset, we observe that while there is a slight performance drop for LightGCN-PGTR at five hops, the overall trend indicates that PGTR still enhances the recommendation performance for longer-range items. This is evident when considering the combined results of Figure 5(c) and (d), where PGTR consistently outperforms LightGCN in capturing long-range collaborative signals.

4.3.2 Case Study. Figure 6 provides a case study on the Amazon-elec dataset, illustrating the recommendation results for user-552. The results show that LightGCN correctly recommends only one item (item-65) at the three-hop distance, whereas LightGCN-PGTR successfully recommends several items that are further away. This case study highlights the capability of PGTR to effectively recommend items that are multiple hops away, demonstrating its superior performance in long-range item recommendation.

4.4 Robustness Study (RQ3)

In this section, we evaluate the robustness of our model compared to other methods against data sparsity and noise.

4.4.1 Robustness against Data Sparsity. In order to evaluate the robustness of PGTR against data sparsity, we conduct various experiments on different sparsity levels. To be more specific, we conduct experiments on different proportions of training sets (20%, 40%, 60%, 80% of the original

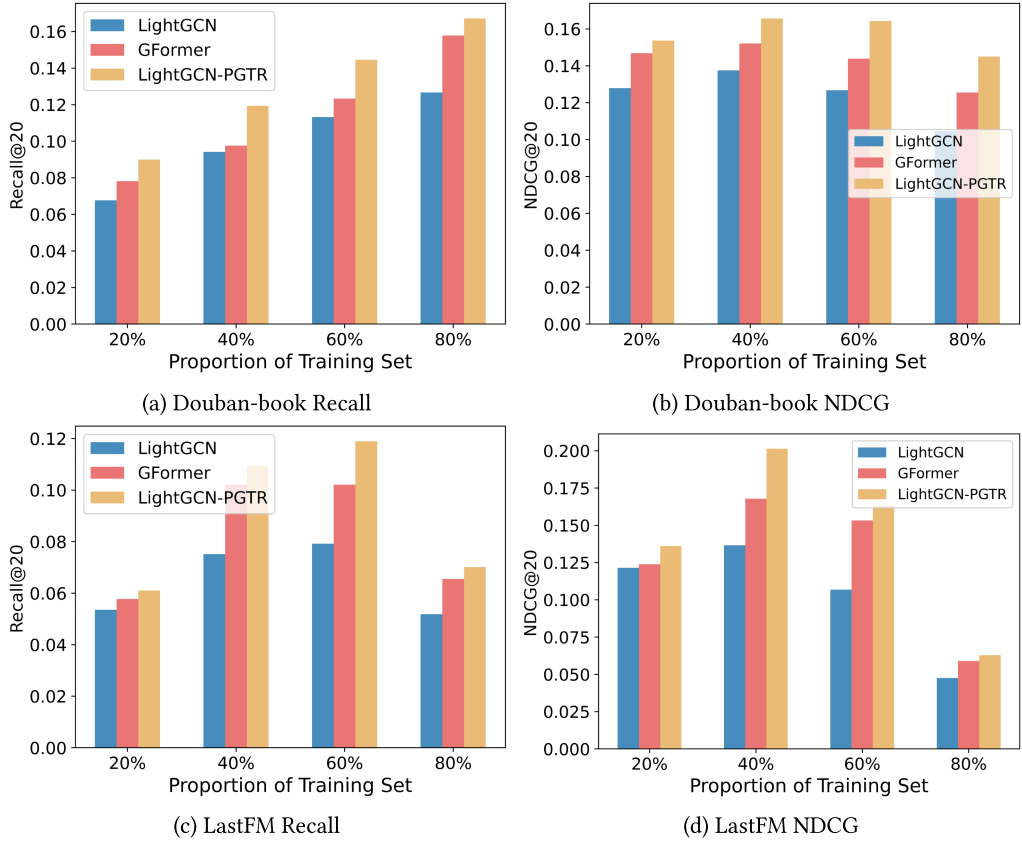


Fig. 7. Performance comparison on Douban-book and LastFM datasets under different proportions of the training set.

dataset, with the remainder used for testing), simulating various levels of sparsity. The results on two datasets, Douban-book and LastFm, are shown in Figure 7. From the figure, it can be observed that our method outperforms LightGCN and GFormer across different levels of sparsity, reaching the SOTA results. We believe that, in the current scenario of sparse recommendation data, adopting PGTR to model relationships among nodes explicitly and capture global collaborative relations is necessary.

4.4.2 Robustness against Data Noise. In order to demonstrate the robustness of the proposed method PGTR against interaction noise, we randomly insert different proportions (10%, 20%, 30%) of noisy interactions into the training set. The noisy interactions of a user are randomly generated based on a specific proportion to the number of interactions. We train PGTR and other powerful baselines and evaluate their performance. We compare the performance decrease of different methods under different proportions of noise. The performance without any noise is the benchmark, as shown in Figure 8. First, we can observe that LightGCN had the highest decrease in performance on both datasets (Amazon-elec and Douban-book). This indicates that adopting this fixed neighbor aggregation is sensitive to noise. Second, it can be observed that SGL and GFormer have a certain ability to resist noise, which demonstrates the robustness of self-supervised learning against noise. Furthermore, the results show that our method has the least performance degradation

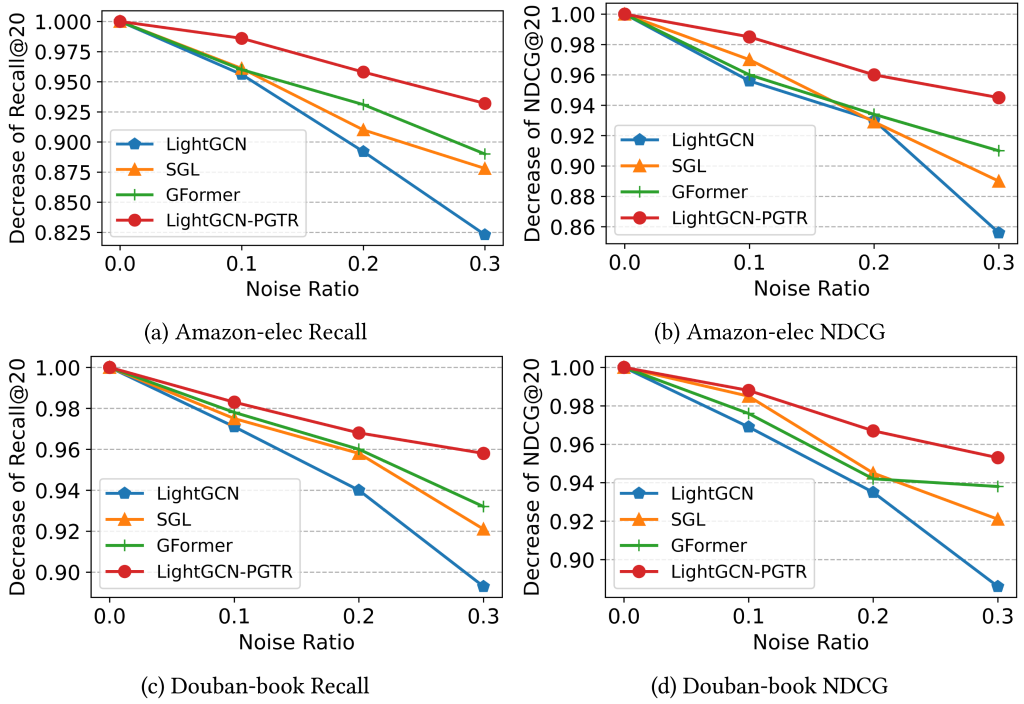


Fig. 8. Performance comparison on Amazon-elec and Douban-book datasets with different noise perturbations.

on both datasets. This demonstrates that our method has good recognition capability for noise when dealing with interaction data. We believe that the PGTR model can integrate multiple positional information to discriminate the authenticity of interactions, which is the source of advancement in our method.

4.5 Ablation Study (RQ4)

In this section, we conduct experiments to validate the effectiveness of the four proposed PEs and the role of capturing long-range/global collaborative information in our model.

4.5.1 Effect of PE. To validate the effectiveness of the proposed PEs, we perform an ablation study over them. Specifically, we separately remove each type of node PE as well as all encodings from the PGTR and test the performance after their removal, i.e., “-PL,” “-DG,” “-PR,” “-TP,” and “All” represent that removing spectral encodings, degree encodings, PageRank encodings, type encodings, and all encodings, respectively. We show the results on Amazon-elec and Douban-book datasets in Table 4. Overall, it can be seen that after removing each PE, the performance would degrade. It means that each PE contributes to the PGTR model. In addition, each PE can have varying effects on different datasets. For example, the impact of spectral encoding on performance is evidently greater in Douban-book compared to its effect on Amazon-elec. We can also observe that the significance of each PE varies. This suggests that they play distinct roles in constructing relationships among nodes across different aspects. As a comparison, the performance drops significantly after all PEs are removed, which indicates that the proposed node PEs play an important role in the model.

4.5.2 Effect of λ_3 . λ_3 is used in Equation (21) to control the ratio of mixing between local and long-range/global collaborative information. We give the tuning results of λ_3 on LastFM and

Table 4. Performance of Ablated Models on Amazon-Elec and Douban-Book Datasets

Dataset	Metric	-PL	-DG	-PR	-TP	-All	PGTR
Amazon-elec	Recall@20	0.0801	0.0777	0.0783	0.0789	0.0613	0.0811
	NDCG@20	0.1089	0.1082	0.1084	0.1084	0.0797	0.1103
Douban-book	Recall@20	0.0873	0.0892	0.0876	0.0889	0.0784	0.0899
	NDCG@20	0.1531	0.1529	0.1512	0.1520	0.1456	0.1536

Bold values indicate the best performance for each metric.

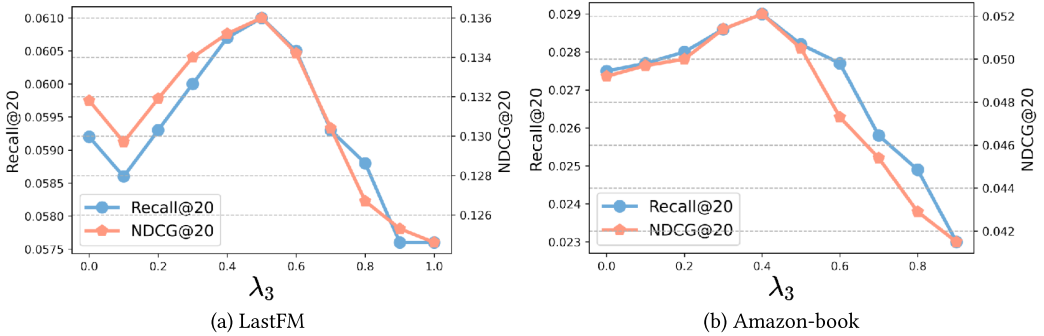


Fig. 9. Effect of λ_3 on LastFM and Amazon-book datasets.

Amazon-book datasets in Figure 9. It demonstrates that global information plays a crucial role in performance. When λ_3 is small, the supplementation of local information is insufficient. When λ_3 is large, introducing excessive global information tends to introduce noise information and thus disrupt node representation learning. These situations can lead to a decrease in performance.

4.5.3 Effect of the SSM Loss. The SSM loss [42] achieves a good performance in recommendation. To further validate the effectiveness of our framework, we compare the performance of the GCN-based model trained with SSM loss on the four datasets, as shown in Table 5. It can be observed that SSM loss significantly improves the performance of the GCN-based model, demonstrating the importance of the self-supervised learning paradigm in recommendation. Furthermore, our approach still outperforms SSM significantly. This indicates that our method more effectively captures users' interests by modeling the relationships among users and items. We do not use the SSM loss on UltraGCN, as UltraGCN has been designed with a specialized loss function tailored to its unique architecture and requirements; thus, there is no comparison with it. It should be noted that in both the “-SSM” and “-PGTR” methods, the temperature coefficient has been searched with the same granularity. Ultimately, the same setting has been applied to ensure a fair and comparable evaluation.

4.6 Computational Efficiency: GPU Utilization and Algorithm Runtime

In the analysis of computational efficiency, as depicted in Tables 6 and 7, the proposed method PGTR demonstrates a tradeoff between computational resources and performance. In this experiment, all methods are trained on an NVIDIA A40 (48 GB) GPU and an Intel Xeon Gold 6326 CPU @

Table 5. Effect of the SSM Loss on the Four Datasets

Data	Amazon-Elec		Douban-Book		LastFM		Amazon-Book	
Model	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
NGCF	0.0684	0.0921	0.0735	0.1164	0.0263	0.0654	0.0141	0.0286
NGCF-SSM	0.0758	0.1035	0.0735	0.1357	0.0587	0.1311	0.0260	0.0469
NGCF-PGTR	0.0802	0.1084	0.0909	0.1558	0.0616	0.1394	0.0274	0.0506
GCCF	0.0358	0.0525	0.0570	0.1114	0.0325	0.0840	0.0179	0.0364
GCCF-SSM	0.0732	0.1030	0.0844	0.1557	0.0447	0.0998	0.0254	0.0457
GCCF-PGTR	0.0789	0.1075	0.0889	0.1584	0.0579	0.1258	0.0264	0.0476
LightGCN	0.0330	0.0485	0.0677	0.1278	0.0535	0.1215	0.0250	0.0472
LightGCN-SSM	0.0613	0.0862	0.0788	0.1465	0.0582	0.1293	0.0273	0.0489
LightGCN-PGTR	0.0811	0.1103	0.0899	0.1536	0.0610	0.1360	0.0290	0.0521

Recall and NDCG scores are reported at a cutoff of 20.

Table 6. Computational Efficiency of PGTR on the LastFM Dataset

LastFM	NGCF	LightGCN	NGCF-PGTR	LightGCN-PGTR
GPU Utilization	2,398 MB	2,238 MB	2,692 MB	2,552 MB
Training Cost (per epoch)	11.5 s	11 s	44 s	44 s
Training Epochs	262	256	48	76
Overall Training Cost	50.2 min	46.9 min	35.2 min	55.7 min
Recall@20	0.0263	0.0535	0.0616	0.0610

2.90 GHz. The training process is terminated if the performance did not improve for 20 consecutive epochs, and the total training time is recorded. When compared to the baseline model LightGCN, LightGCN-PGTR exhibits a moderate increase in GPU utilization, with 2,552 MB and 2,972 MB on the LastFM and Amazon-book datasets, respectively. This is expected due to the additional PE modules and the Transformer module's complexity. However, the single-round training time for LightGCN-PGTR (NGCF-PGTR) is approximately $5\times$ that of LightGCN (NGCF-PGTR), which is attributed to the more complex model architecture and the computational overhead of capturing long-range dependencies. Despite this, the overall training duration of Light-PGTR is only $1.2\times$ and $2.9\times$ longer on the LastFM and Amazon-book datasets, respectively, when compared to LightGCN. Specifically, NGCF-PGTR even has a shorter overall training time than NGCF on the LastFM dataset. This indicates that while PGTR requires more computational resources and time per epoch, it converges faster, requiring fewer epochs to achieve greater performance. This efficiency in terms of epochs is crucial for large-scale recommendation tasks where the volume of data is extensive, and the model must generalize well from limited training data. Overall, the increased resource usage of PGTR is justified by its improved ability to capture long-range collaborative signals, leading to enhanced recommendation accuracy and robustness against sparsity and noise.

Table 7. Computational Efficiency of PGTR on the Amazon-Book Dataset

Amazon-Book	NGCF	LightGCN	NGCF-PGTR	LightGCN-PGTR
GPU Utilization	2,676 MB	2,376 MB	3,188 MB	2,972 MB
Training Cost (per epoch)	13 s	11.5 s	59 s	67 s
Training Epochs	118	254	81	128
Overall Training Cost	25.6 min	48.7 min	79.7 min	142.9 min
Recall@20	0.0141	0.0250	0.0274	0.0290

5 Related Work

In this section, we review related works that include graph-based recommendation and PEs of GT.

5.1 Graph-based Recommendation

Graph-based recommendation approaches have gained attention for effectively capturing complex relationships among users and items, making them suitable for personalized recommendations [7, 8, 44]. These methods leverage the inherent structure and connectivity of data to enhance recommendation performance. Various algorithms like NGCF [39], LightGCN [15], UltraGCN [27], and so on exploit the graph's information for recommendation tasks. NGCF [39] utilizes graph convolutional networks to capture the complex patterns in user-item interactions. It propagates user and item embeddings through the graph to refine their representations, which are then used for making recommendations. LightGCN [15] is a simplified version of GCN designed specifically for recommendation tasks. It removes complex operations such as feature transformation and nonlinear activation, focusing solely on the essential component of neighborhood aggregation. This results in a model that is not only easier to train but also achieves significant improvements in recommendation performance over more complex models like NGCF. UltraGCN [27] is another advancement in graph-based recommendations. It introduces an ultra-high-order proximity preserving scheme that captures long-range dependencies in the user-item graph. This allows the model to better understand the intricate connections between users and items, leading to enhanced recommendation accuracy. However, these methods are constrained by local information and lack direct and explicit modeling of long-range collaborative signals, which can lead to performance loss. Recently, there have been several studies aiming to encode global information into local representations based on contrastive learning [3, 46, 49]. By this way, nodes can obtain long-range collaborative signals. For example, EGLN [49] proposes a local-global consistency optimization function to capture the global properties learned from the augmented interaction graph, which is obtained by calculating the similarities between users and items, retaining several largest similarities [52]. This approach allows the model to better understand the overall structure of the user-item interactions, leading to improved recommendations even for users and items with sparse interaction histories. LightGCL [3] utilizes the principle components of the interaction graph to distill important collaborative signals from the global perspective and then refines node representations by local-global contrastive learning. This method is particularly effective in scenarios where the interaction graph is dense and high-dimensional, as it can capture the underlying patterns that are not immediately apparent from local connections alone. However, these methods heavily rely on the way of constructing global collaborative relations and may lose some important signals. The construction of the augmented interaction graph in EGLN [49], for instance, depends on the selection of top similarities, which can be sensitive to the choice of threshold and may exclude some valuable interactions. Similarly,

LightGCL [3] depends on the accurate identification of principal components, which may not always capture the most relevant aspects of the global structure, especially in cases where the interaction graph is noisy or incomplete.

Additionally, auxiliary information such as item attributes, user demographics, and knowledge graph is incorporated into the graph representation to capture diverse preferences and improve recommendation accuracy [4, 25, 38]. Recently, self-supervised graph-based methods have been widely explored and utilized in the field of recommendation systems [26, 41, 46, 48, 51]. They facilitate the discovery of meaningful structures and embeddings within the graph, leading to improved recommendations.

The GT has gained popularity in multimedia recommendation [40], multi-behavior recommendation [45], and sequential recommendation [13]. However, its exploration in the field of collaborative filtering remains limited. GFormer [23] and SHT [47] utilize the GT for global relations modeling. However, the modeling is not adequate. In this work, we explore using the GT to capture collaborative relationships between all pairs of nodes, which is different from them.

5.2 PE of GT

PE is a critical technique in GT that embeds positional information into the model, allowing it to understand the relative positions of nodes within a graph. This is essential because the transformer architecture, which GTs are based on, does not inherently account for order or position. Various forms of PE have been explored to enhance the performance of GTs in recommendation systems.

Laplacian PE is one such method that uses the graph's Laplacian matrix to capture the structural properties of the graph [12, 22]. This encoding helps in understanding the role of each node within the graph's structure, providing a sense of the node's "position" in terms of its connectivity to other nodes. Shortest-path-distance encoding is another approach that considers the shortest path distances between nodes as a measure of their relative positions [24]. This is particularly useful in recommendation systems where the goal is to recommend items that are "close" to the user's current preferences, even if they are not directly connected. Centrality-based encoding leverages the concept of node centrality, which measures the importance or influence of a node within the graph [50]. Nodes with high centrality are likely to be more influential in shaping the recommendations, as they are more connected to other nodes. In the context of collaborative filtering, PE can help in distinguishing between nodes that have similar interaction patterns but different positions within the graph. For example, two users may have similar preferences, but their positions in the social graph or their influence within the community might differ, affecting the recommendations they receive.

To the best of our knowledge, there has been limited work on utilizing PE in collaborative filtering scenarios. GFormer [23], in particular, only uses the distance between the target node and sampled anchor nodes to preserve global topological structure information. We argue that this is insufficient. Exploring how to effectively encode positional information in collaborative filtering is a new question. In this article, we design several simple yet effective PEs tailored for the user-item interaction graph in collaborative recommendation.

In summary, PE in GTs for recommendation systems is an evolving field with promising potential. By providing explicit positional information, these encodings help models capture the intricate spatial dependencies within user-item graphs, leading to more personalized and accurate recommendations.

6 Conclusion and Future Work

In this article, our focus is incorporating the GT into collaborative recommendation to address an important problem existing in the GCN-based methods: capturing long-range collaborative signals. To this end, we propose the PGTR framework which can be applied to various GCN-based backbones.

The key contribution of this study lies in the design of node PEs tailored for the interaction graph in collaborative recommendation within PGTR for learning relationships between nodes. We conduct experiments on multiple datasets, demonstrating the effectiveness of our approach.

In the future, we will design more effective PEs for different interaction graphs, exploring how to capture graph structural information and a variety of node relationships. Furthermore, we will explore the application of GT in various recommendation scenarios, addressing issues of data sparsity and node dependency in recommendations.

References

- [1] Uri Alon and Eran Yahav. 2020. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*.
- [2] Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *AAAI Conference on Artificial Intelligence*, Vol. 34, 7464–7471.
- [3] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2022. LightGCL: Simple yet effective graph contrastive learning for recommendation. In *The 11th International Conference on Learning Representations*.
- [4] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The World Wide Web Conference*, 151–161.
- [5] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI Conference on Artificial Intelligence*, Vol. 34, 3438–3445.
- [6] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. 2022. NAGphormer: A tokenized graph transformer for node classification in large graphs. In *The 11th International Conference on Learning Representations*.
- [7] Jiajia Chen, Jiancan Wu, Jiawei Chen, Xin Xin, Yong Li, and Xiangnan He. 2024. How graph convolutions amplify popularity bias for recommendation? *Frontiers of Computer Science* 18, 5 (2024), 185603.
- [8] Jiajia Chen, Xin Xin, Xianfeng Liang, Xiangnan He, and Jun Liu. 2022. GDSRec: Graph-based decentralized collaborative filtering for social recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2022), 4813–4824.
- [9] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *AAAI Conference on Artificial Intelligence*, Vol. 34, 27–34.
- [10] Krzysztof Marcin Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. In *International Conference on Learning Representations*.
- [11] Aaron Clauset, Cosma Rohilla Shalizi, and Mark E. J. Newman. 2009. Power-law distributions in empirical data. *SIAM Review* 51, 4 (2009), 661–703.
- [12] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. arXiv:2012.09699. Retrieved from <https://arxiv.org/abs/2012.09699>
- [13] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S. Yu. 2021. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *30th ACM International Conference on Information & Knowledge Management*, 433–442.
- [14] Ziwei Fan, Ke Xu, Zhang Dong, Hao Peng, Jiawei Zhang, and Philip S. Yu. 2023. Graph collaborative signals denoising and augmentation for recommendation. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2037–2041.
- [15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 639–648.
- [16] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2354–2366.
- [17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*, San Juan, Puerto Rico, May 2–4, 2016. Conference Track Proceedings. Yoshua Bengio and Yann LeCun (Eds.).
- [18] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. 2022. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems*, Vol. 35, 14582–14595.
- [19] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.
- [20] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 426–434.

- [21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
- [22] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems*, Vol. 34, 21618–21629.
- [23] Chaoliu Li, Lianghao Xia, Xubin Ren, Yaowen Ye, Yong Xu, and Chao Huang. 2023. Graph transformer for recommendation. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Pobleto (Eds.), ACM, 1680–1689. DOI: <https://doi.org/10.1145/3539618.3591723>
- [24] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *Advances in Neural Information Processing Systems*, Vol. 33, 4465–4478.
- [25] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *28th ACM International Conference on Information and Knowledge Management*, 539–548.
- [26] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *ACM Web Conference 2022*, 2320–2329.
- [27] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra simplification of graph convolutional networks for recommendation. In *Conference on Information and Knowledge Management*.
- [28] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. 2022. Transformer for graphs: An overview from architecture perspective. arXiv:2202.08455. Retrieved from <https://arxiv.org/abs/2202.08455>
- [29] Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampásek. 2023. Attending to graph transformers. arXiv:2302.04181. Retrieved from <https://arxiv.org/abs/2302.04181>
- [30] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. 2022. Universal graph transformer self-attention networks. In *Companion Proceedings of the Web Conference 2022*, 193–196.
- [31] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. *The Pagerank Citation Ranking: Bring Order to the Web*. Technical Report. Stanford University.
- [32] Wonpyo Park, Woong-Gi Chang, Donggeon Lee, Juntae Kim, and Seung-won Hwang. 2022. GRPE: Relative positional encoding for graph transformer. In *ICLR2022 Machine Learning for Drug Discovery*.
- [33] Ladislav Rampásek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2022. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, Vol. 35, 14501–14515.
- [34] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.
- [35] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, Montreal, QC, Canada, June 18–21, 2009. Jeff A. Bilmes and Andrew Y. Ng (Eds.), AUAI Press, 452–461.
- [36] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 635–644.
- [37] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. In *Advances in Neural Information Processing Systems*, Vol. 33, 12559–12571.
- [38] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *The World Wide Web Conference*, 3307–3313.
- [39] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 165–174.
- [40] Yinwei Wei, Wenqi Liu, Fan Liu, Xiang Wang, Liqiang Nie, and Tat-Seng Chua. 2023. LightGT: A light graph transformer for multimedia recommendation. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1508–1517.
- [41] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 726–735.
- [42] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, Tianyu Qiu, and Xiangnan He. 2024. On the effectiveness of sampled softmax loss for item recommendation. *ACM Transactions on Information Systems* 42, 4 (2024), 1–26.
- [43] Qitian Wu, Wentao Zhao, Zenan Li, David P. Wipf, and Junchi Yan. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems*, Vol. 35, 27387–27401.

- [44] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: A survey. *ACM Computing Surveys* 55, 5 (2022), 1–37.
- [45] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. 2021. Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In *AAAI Conference on Artificial Intelligence*, Vol. 35, 4486–4493.
- [46] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 70–79.
- [47] Lianghao Xia, Chao Huang, and Chuxu Zhang. 2022. Self-supervised hypergraph transformer for recommender systems. In *28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2100–2109.
- [48] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge graph contrastive learning for recommendation. In *45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1434–1443.
- [49] Yonghui Yang, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2021. Enhanced graph learning for collaborative filtering via mutual information maximization. In *44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 71–80.
- [50] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, Vol. 34, 28877–28888.
- [51] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? Simple graph contrastive learning for recommendation. In *45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1294–1303.
- [52] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2023. Self-supervised learning for recommender systems: A survey. *IEEE Transactions on Knowledge and Data Engineering* 36, 1 (2023), 335–355.
- [53] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. arXiv:2001.05140. Retrieved from <https://arxiv.org/abs/2001.05140>
- [54] Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. 2021. Gophormer: Ego-graph transformer for node classification. arXiv:2110.13094. Retrieved from <https://arxiv.org/abs/2110.13094>

Received 16 October 2024; revised 13 April 2025; accepted 9 July 2025