# Rankformer: A Graph Transformer for Recommendation based on Ranking Objective

Sirui Chen[†][‡]
Zhejiang University
Hangzhou, China
chenthree@zju.edu.cn

Shen Han[‡]
Zhejiang University
Hangzhou, China
22421249@zju.edu.cn

Jiawei Chen[*][†][‡][§]
Zhejiang University
Hangzhou, China
sleepyhunt@zju.edu.cn

Binbin Hu
Ant Group
Hangzhou, China
bin.hbb@antfin.com

Sheng Zhou
Zhejiang University
Hangzhou, China
zhousheng_zju@zju.edu.cn

Gang Wang
Bangsun Technology
Hangzhou, China
wanggang@bsfit.com.cn

Yan Feng[†][‡]
Zhejiang University
Hangzhou, China
fengyan@zju.edu.cn

Chun Chen[†][‡]
Zhejiang University
Hangzhou, China
chenc@zju.edu.cn

Can Wang[†][§]
Zhejiang University
Hangzhou, China
wcan@zju.edu.cn

## Abstract

Recommender Systems (RS) aim to generate personalized ranked lists for each user and are evaluated using ranking metrics. Although personalized ranking is a fundamental aspect of RS, this critical property is often overlooked in the design of model architectures. To address this issue, we propose Rankformer, a ranking-inspired recommendation model. The architecture of Rankformer is inspired by the gradient of the ranking objective, embodying a unique (graph) transformer architecture — it leverages global information from all users and items to produce more informative representations and employs specific attention weights to guide the evolution of embeddings towards improved ranking performance. We further develop an acceleration algorithm for Rankformer, reducing its complexity to a linear level with respect to the number of positive instances. Extensive experimental results demonstrate that Rankformer outperforms state-of-the-art methods. The code is available at https://github.com/StupidThree/Rankformer.

## CCS Concepts

• **Information systems → Recommender systems**.

## Keywords

Recommendation; Transformer; Personalized Ranking

[*]Corresponding author.
[†]State Key Laboratory of Blockchain and Data Security, Zhejiang University.
[‡]College of Computer Science and Technology, Zhejiang University.
[§]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security.

**ACM Reference Format:**
Sirui Chen, Shen Han, Jiawei Chen, Binbin Hu, Sheng Zhou, Gang Wang, Yan Feng, Chun Chen, and Can Wang. 2025. Rankformer: A Graph Transformer for Recommendation based on Ranking Objective. In *Proceedings of the ACM Web Conference 2025 (WWW '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3696410.3714547

## 1 Introduction

Recommender systems (RS) have been integrated into many personalized services, playing an essential role in various applications [29]. A fundamental attribute that distinguishes RS from other machine learning tasks is its inherently *personalized ranking* nature. Specifically, RS aims to create user-specific ranked lists of items and retrieve the most relevant ones for recommendation [45]. To achieve this, most existing RS approaches adopt model-based paradigms, where a recommendation model is learned from users' historical interactions and subsequently generates ranking scores for each user-item pair.

Recent years have witnessed substantial progress in recommendation model architectures, evolving from basic matrix factorization [30, 38] to more advanced architectures like Auto-Encoder [15, 34] and Graph Neural Networks (GNNs) [18, 55, 56, 67]. Despite their increasingly sophisticated, we argue that a critical limitation remains — **their architecture design often neglects the essential ranking property of RS, which could compromise their effectiveness.**

To illustrate this point, let's take GNN-based architectures as examples for analyses. GNN-based models have been extensively studied in this field and usually achieve state-of-the-art performance [26]. In RS, the primary role of GNNs has been identified as a low-pass filter (*a.k.a.* graph smoother) [42], which draws the embeddings of connected users/items closer. However, this role significantly deviates from the ranking objective, thereby creating a gap that can impede their effectiveness. As demonstrated in Figure 1, without the guidance of a ranking objective function,

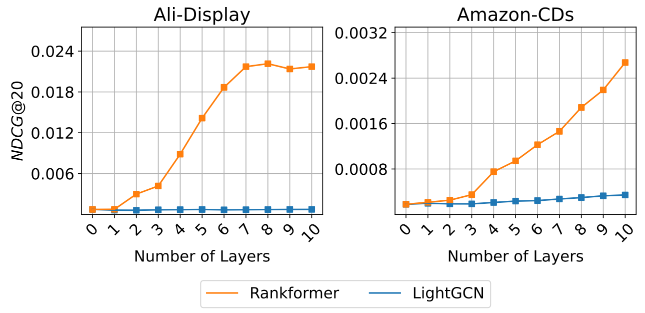

**Figure 1: Performance in terms of $NDCG@20$ when using Rankformer and LightGCN with different numbers of layers for randomly initialized representations without training.**

the performance gains from stacking multiple layers of GNNs are limited. Even when supervised signals are introduced, stacking multiple GNN layers would easily result in the notorious issue of over-smoothing, where the score differences between positive and negative instances are reduced and become indistinguishable. This is in direct conflict with the ranking objective, which aims to elevate scores of the positive instances over the negative ones, leading to suboptimal performance. These limitations clearly underscore the necessity of considering the ranking property in the design of model architectures. It can serve as an *inductive bias* to guide the model towards generating better ranking performance. Naturally, a significant research question arises: **How can we develop a recommendation architecture that is well-aligned with the ranking property of RS?**

Towards this end, we propose a novel ranking-inspired recommendation model, **Rankformer**. Instead of a heuristic design or inheriting from other domains, the architecture of Rankformer is directly inspired by the ranking objective. Specifically, we scrutinize the gradient of the ranking objective, which suggests the evolution direction of embeddings for enhanced ranking performance, and consequently develop a neural layer in accordance with this gradient. Rankformer embodies a unique (graph) transformer architecture: It leverages global information from all other users and items to obtain more informative user/item representations. Furthermore, specific attention weights, which compare positive and negative instances, are introduced to guide the evolution of embeddings towards improved ranking performance.

Despite its theoretical effectiveness, the implementation of Rankformer could encounter the challenge of computational inefficiency. As a transformer model, Rankformer involves information propagation between each user and item, leading to a time complexity that is quadratic with respect to the number of users and items. This renders it computationally prohibitive in practice. Thus, we propose an acceleration algorithm specifically tailored for Rankformer. By utilizing mathematical transformations and memorization skills, we reduce the complexity to linear with respect to the number of positive instances, significantly improving the model's efficiency.

Overall, this work makes the following contributions:

- Proposing a novel ranking-inspired recommendation model, Rankformer, that explicitly incorporates the personalized ranking principle into the model architecture design.
- Customizing a fast algorithm for Rankformer, reducing the original computational complexity from quadratic with the number of users and items to linear with the positive instances.
- Conducting experiments on four real-world datasets to demonstrate the superiority of Rankformer over state-of-the-art methods by a significant margin (4.48% on average).

## 2 Preliminary

In this section, we present the background of recommender systems and the transformer model.

### 2.1 Background on Recommender Systems

**Task Formulation.** This work focuses on collaborative filtering, a generic and common recommendation scenario. Given a recommender system with a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$, let $n$ and $m$ denote the number of users and items. Let $\mathcal{D} = \{y_{ui} : u \in \mathcal{U}, i \in \mathcal{I}\}$ denote the historical interactions between users and items, where $y_{ui} = 1$ indicates that user $u$ has interacted with item $i$, and $y_{ui} = 0$ indicates has not. For convenience, we define $\mathcal{N}_u^+ = \{i \in \mathcal{I} : y_{ui} = 1\}$ as the set of positive items for user $u$, and $\mathcal{N}_u^- = \mathcal{I} \setminus \mathcal{N}_u^+$ as the negative item set. Similar definitions apply for $\mathcal{N}_i^+$ and $\mathcal{N}_i^-$ which indicate the set of positive and negative users for item $i$, respectively. We also define $d_u = |\mathcal{N}_u^+|$ and $d_i = |\mathcal{N}_i^+|$ are the number of positive items/users for user $u$/item $i$. The recommendation task can be formulated as learning a personalized ranked list of items for users and recommending the top items that users are most likely to interact with.

Personalized ranking is a fundamental property that distinguishes RS from other machine-learning tasks. RS is typically evaluated using ranking metrics such as NDCG@K, AUC, and Precision@K, which measure how well positive instances are ranked higher than negative ones [58]. Given this, the importance of considering the ranking property in model architecture design cannot be overstated.

**Recommendation Models.** Modern recommender systems typically model ranking scores using a learnable recommendation model. Embedding-based models are widely adopted [30, 45]. These models map the features (*e.g.*, ID) of users and items into a $d$-dim embedding $\mathbf{z_u}, \mathbf{z_i} \in \mathbb{R}^d$, and generate their predicted score $\hat{y}_{ui}$ based on similarity of embeddings. The inner product, inherited from matrix factorization (MF), is commonly used in RS, *i.e.*, $\hat{y}_{ui} = \mathbf{z}_u^T \mathbf{z}_i$, as it supports efficient retrieval and has demonstrated strong performance in various scenarios [2, 26]. The predicted scores $\hat{y}_{ui}$ are subsequently utilized to rank items for generating recommendations. For convenience, we collect the embeddings of all users and items as a matrix $\mathbf{Z}$.

Given the ranking nature of RS, existing recommendation models are often trained with ranking-oriented objective functions, *e.g.*, BPR [45]. BPR aims to raise the scores of positive items relative to negative ones and can be formulated as:

$$\mathcal{L}_{BPR} = -\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{N}_u^+} \sum_{j \in \mathcal{N}_u^-} \frac{\sigma(\mathbf{z}_u^T \mathbf{z}_i - \mathbf{z}_u^T \mathbf{z}_j)}{d_u(m - d_u)} + \lambda \|\mathbf{Z}\|_2^2 \tag{1}$$

where $\sigma(.)$ denotes the activation, and the hyperparameter $\lambda$ controls the strength of the regularizer.

While BPR provides a supervised signal to guide model training toward better ranking, it does not negate the importance of model architecture. As demonstrated in recent machine learning literature [7, 8, 74], model architecture acts as an inductive bias that determines the model's learning and generalization capacity — if the bias aligns well with the underlying patterns of the task, the model is likely to perform well. In subsection 3.2, we conduct specific analyses to demonstrate the merits of incorporating ranking properties into model architecture design.

**GNN-based Recommendation Models.** In recent years, Graph Neural Networks (GNNs) have been widely explored in the field of RS and have demonstrated notable effectiveness [18, 25, 26, 56]. These methods construct a bipartite graph from historical interactions, where users and items are represented as nodes and an edge exists between them if the user has interacted with the item. User/item embeddings are iteratively refined by aggregating information from their graph neighbors. Formally, taking the representative LightGCN [26] model as an example, the $l$-layer network can be written as:

$$\mathbf{z}_u^{(l+1)} = \sum_{i \in \mathcal{N}_u^+} \frac{1}{\sqrt{d_u d_i}} \mathbf{z}_i^{(l)}; \quad \mathbf{z}_i^{(l+1)} = \sum_{u \in \mathcal{N}_i^+} \frac{1}{\sqrt{d_u d_i}} \mathbf{z}_u^{(l)} \tag{2}$$

Recent work has shown that GNNs act as low-pass filters, which is beneficial for capturing collaborative signals [71]. Nevertheless, as discussed earlier, there is a gap between the role of GNNs and the ranking objective, limiting their effectiveness. While the work [47] attempts to build theoretical connections, their finding is constrained by impractical assumptions, such as requiring large embedding spaces, single-layer GNNs, or untrained models.

## 2.2 Transformer Architecture

Transformer has been widely adopted in various fields [5, 48, 50]. The core of the transformer is the attention module, which takes input $X \in \mathbb{R}^{(n+m)\times d}$ and computes:

$$\begin{aligned} &\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V, \\ &\text{Attn}(\mathbf{X}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}})\mathbf{V} \end{aligned} \tag{3}$$

where $\mathbf{W}_Q \in \mathbb{R}^{d\times d_K}, \mathbf{W}_K \in \mathbb{R}^{d\times d_K}, \mathbf{W}_V \in \mathbb{R}^{d\times d_V}$ are weight matrices for the query, key, and value projections, respectively.

The transformer architecture has significant potential to be exploited in RS. Considering the input as embeddings of users and items, the transformer estimates the similarity between these entities based on their projected embeddings and then aggregates information from other entities according to this similarity. Such operations align closely with the fundamental principles of collaborative filtering [6, 32]. However, the architecture should be specifically designed to align with the ranking principle, and the high computational overhead from global aggregation needs to be addressed.

## 3 Methodology

In this section, we first introduce the architecture of the proposed Rankformer, followed by a discussion of its connections with existing architectures. Finally, we detail how its computation is accelerated.

## 3.1 Rankformer Layer

Rankformer is directly inspired by the ranking objective — we aim to promote user/item embeddings to evolve towards better ranking performance as they progress through the neural network. To achieve this, a natural idea is to let the embeddings evolve in alignment with the gradient of the ranking objective, which suggests the directions for enhancing ranking performance. Specifically, let $\mathcal{L}(\mathbf{Z})$ be a specific ranking objective. We may employ gradient descent (or ascent) to update the embeddings to improve their quality with:

$$\mathbf{Z}^{(l+1)} = \mathbf{Z}^{(l)} - \tau \cdot \frac{\partial \mathcal{L}(\mathbf{Z}^{(l)})}{\partial \mathbf{Z}^{(l)}} \tag{4}$$

where $\mathbf{Z}^{(l+1)}$ denotes the embeddings in the $l$-th step and $\tau$ denotes the step-size. This inspires us to develop a neural network that mirrors such an update. We may simply let $\mathbf{Z}^{(l+1)}$ be the embeddings of the $l$-layer and employ a neural network along with Eq(1). Naturally, the neural network would capture the ranking signals of the recommendation and guide the embeddings to evolve toward better ranking performance.

To instantiate this promising idea, this work simply adopts the conventional objective BPR for neural network design, as it has been demonstrated to be an effective surrogate for the AUC metric. Moreover, for facilitating analyses and accelerating computation, we simply choose the quadratic function activation, *i.e.*, $\sigma(x) = x^2 + cx$. This can also be considered as a second-order Taylor approximation of other activations. The $l$-layer neural network of Rankformer can be formulated as follows by mirroring the gradient descent of BPR:

$$\begin{aligned} \mathbf{z}_u^{(l+1)} &= (1-\tau)\mathbf{z}_u^{(l)} + \frac{\tau}{C_u^{(l)}} \left( \underbrace{\sum_{i\in\mathcal{N}_u^+} \Omega_{ui}^{+\,(l)} \mathbf{z}_i^{(l)}}_{\text{Aggregate Positive}} + \underbrace{\sum_{i\in\mathcal{N}_u^-} \Omega_{ui}^{-\,(l)} \mathbf{z}_i^{(l)}}_{\text{Aggregate Negative}} \right) \\ \mathbf{z}_i^{(l+1)} &= (1-\tau)\mathbf{z}_i^{(l)} + \frac{\tau}{C_i^{(l)}} \left( \underbrace{\sum_{u\in\mathcal{N}_i^+} \Omega_{iu}^{+\,(l)} \mathbf{z}_u^{(l)}}_{\text{Aggregate Positive}} + \underbrace{\sum_{u\in\mathcal{N}_i^-} \Omega_{iu}^{-\,(l)} \mathbf{z}_u^{(l)}}_{\text{Aggregate Negative}} \right) \end{aligned} \tag{5}$$

where $\Omega^+$ and $\Omega^-$ denote the weights for aggregating positive and negative users/items by items/users:

$$\begin{aligned} \Omega_{ui}^{+\,(l)} = \Omega_{iu}^{+\,(l)} &= \frac{1}{d_u} \left( \underbrace{(\mathbf{z}_u^{(l)})^T \mathbf{z}_i^{(l)}}_{\text{Similarity}} - \underbrace{b_u^{-\,(l)}}_{\text{Benchmark}} + \underbrace{\alpha}_{\text{Offset}} \right) \\ \Omega_{ui}^{-\,(l)} = \Omega_{iu}^{-\,(l)} &= \frac{1}{m-d_u} \left( \underbrace{(\mathbf{z}_u^{(l)})^T \mathbf{z}_i^{(l)}}_{\text{Similarity}} - \underbrace{b_u^{+\,(l)}}_{\text{Benchmark}} - \underbrace{\alpha}_{\text{Offset}} \right) \end{aligned} \tag{6}$$

where $b_u^{+(l)}$ and $b_u^{-(l)}$ are two benchmark terms used to calculate the average similarity of positive/negative pairs.

$$\begin{aligned} b_u^{+(l)} &= \frac{1}{d_u}\sum_{i\in\mathcal{N}_u^+}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} \\ b_u^{-(l)} &= \frac{1}{m-d_u}\sum_{i\in\mathcal{N}_u^-}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} \end{aligned} \tag{7}$$

Detailed derivations can be found in Appendix A.1. Here we simply set the hyperparameter $\lambda = 1$. Additionally, we introduce normalization constants to maintain numerical stability, *i.e.,*

$$\begin{aligned} C_u^{(l)} &= \sum_{i\in\mathcal{N}_u^+}\left|\Omega_{ui}^{+(l)}\right| + \sum_{i\in\mathcal{N}_u^-}\left|\Omega_{ui}^{-(l)}\right| \\ C_i^{(l)} &= \sum_{u\in\mathcal{N}_i^+}\left|\Omega_{iu}^{+(l)}\right| + \sum_{u\in\mathcal{N}_i^-}\left|\Omega_{iu}^{-(l)}\right| \end{aligned} \tag{8}$$

While the neural layer may seem complex, its underlying intuition is straightforward. The derived Rankformer embodies a unique transformer architecture—each user (or item) iteratively leverages global information from all items (or users) to update its representation, utilizing both positive and negative interactions. For example, a positive item indicates what the user likes, while a negative item suggests what the user dislikes. Both signals are crucial for profiling user preferences. Aggregating information from both types of items helps guide the user's embedding closer to the items they like and away from the items they dislike. Similar logic applies to the item side, where aggregating information from both positive and negative users enhances the quality of item embeddings.

Furthermore, specific attention weights (*i.e.,* , Eq(6)) are introduced, consisting of three parts:

- Similarity term $\mathbf{z}_u^T\mathbf{z}_i$: This operation can be seen as a vanilla attention mechanism akin to that in the transformer model, except that we omit the extra projection parameters. The intuition here is that similar entities provide more valuable information and thus should be given higher attention weights during information aggregation.
- Benchmark term $b_u^+$ or $b_u^-$: Rankformer incorporates additional benchmark terms that represent the average similarity among all positive (or negative) items for each user. This approach aligns with the inherent ranking nature of RS. The absolute similarity value (*i.e.,* the prediction score) may not always be the most important factor; rather, the relative score — *i.e.,* how much an item's score is higher (or lower) compared to others — provides crucial evidence of its ranking, indicating the degree of its positivity or negativity. Thus, for positive instances, a negative benchmark is used as a reference point for comparison, and a similar strategy is applied to negative instances.
- Offset term $\alpha$: This term differentiates the influence of positive and negative interactions. For positive user-item pairs, the weights are increased, bringing their embeddings closer, while for negative pairs, the weights are reduced (and can even become negative), pushing their embeddings apart. Additionally, the magnitude of $\alpha$ can act as a smoothing factor. A larger $\alpha$ makes the weight distribution among positive (or negative) instances more uniform while a smaller $\alpha$ sharpens the distribution.

Table 1: Comparison of Rankformer with LightGCN, GAT, and vanilla Transformer. These three methods can be viewed as special cases of Rankformer with certain components removed.

| | $\Omega_{ui}^+$ | $\Omega_{ui}^-$ |
|---|---|---|
| LightGCN | $\frac{1}{d_u}$ | 0 |
| GAT | $\frac{\mathbf{z}_u^T\mathbf{z}_i}{d_u}$ | 0 |
| Transformer | $\frac{\mathbf{z}_u^T\mathbf{z}_i}{d_u}$ | $\frac{\mathbf{z}_u^T\mathbf{z}_i}{d_u}$ |
| Rankformer | $\frac{\mathbf{z}_u^T\mathbf{z}_i-b_u^-+\alpha}{d_u}$ | $\frac{\mathbf{z}_u^T\mathbf{z}_i-b_u^--\alpha}{d_u}$ |

## 3.2 Disscussion

In this subsection, we elucidate the distinctions and connections between Rankformer and existing methodologies. Table 1 summarizes these relationships, wherein LightGCN, GAT, and the vanilla Transformer can be viewed as special cases of Rankformer with certain components omitted.

**Comparison with GNNs-based methods:** When compared to existing GNNs-based methods (*e.g.,* LightGCN[26], GAT [51]), the primary distinction of Rankformer is its ability to leverage signals from negative instances during the aggregation process. This aspect is pivotal, as negative instances also provide valuable information for profiling user preferences or item attributes. For instance, negative items supply signals about user dislikes, which are equally valuable for learning user representations, *i.e.,* distancing the user's representation from the item.

**Comparison with Transformer:** When compared to existing (graph) transformer models [18, 26], Rankformer exhibits three differences: 1) During information aggregation, Rankformer handles positive and negative relations separately, calculating their attention weights in different manners. This treatment is rational, as they deliver distinctly different types of signals, enabling the transformer to perceive such crucial historical interaction information. 2) Beyond embedding similarity, Rankformer introduces additional benchmark and offset terms, ensuring the neural network aligns well with the ranking objective. 3) Most importantly, Rankformer is not heuristically designed but is entirely derived from the ranking objective, guiding the evolution of embeddings toward improved ranking performance.

**Theoretical advantage of Rankformer:** Given this, some readers may question the necessity of simulating gradient descent of ranking objectives through the design of the model architecture. To demonstrate the effectiveness of Rankformer, we conduct the following theoretical analyses, which connect the Rankformer with the Levenberg-Marquardt algorithm [44].

Specifically, let $\mathcal{L}(\theta)$ be the loss function of a model (i.e., BPR), and $\theta$ be the parameters of the model. The architecture of Rankformer is designed to simulate multiple iterations of gradient descent from the original embeddings, each layer of Rankformer can be formulated as $\mathcal{L}_R(\theta) = \mathcal{L}(\theta - \tau\nabla\mathcal{L}(\theta))$, where $\tau$ denotes the step-size, and $\theta - \tau\nabla\mathcal{L}(\theta)$ denotes the gradient descent that Rankformer simulates. Intuitively, Rankformer looks one step ahead, targeting the reduction of loss under a one-step gradient descent. This approach offers a better optimization direction than directly optimizing BPR Loss. The lemma below elucidates the theoretical connection between Rankformer and the Levenberg-Marquardt algorithm, which is known for faster convergence, improved stability, and enhanced performance [14, 37, 41]:

Lemma 3.1. *Performing gradient descent on $\mathcal{L}_R(\theta)$ is equivalent to using the Levenberg-Marquardt algorithm to perform gradient descent on $\mathcal{L}(\theta)$.*

The proof is presented in Appendix A.2. From the lemma, we find that the forward-looking mechanism adopted in Rankformer to reduce the loss under one-step gradient descent can be theoretically approximated to the Levenberg–Marquardt algorithm, implicitly utilizing second-order derivative information, providing a better optimization direction.

## 3.3 Fast Implementation

As a transformer model, the implementation of Rankformer would also face an inefficiency challenge. The computational overhead primarily originates from its information aggregation mechanism, which involves global aggregation between users and items, with the complexity $O(nmd)$. Given the extensive number of users and items in real-world scenarios, such operations can be prohibitively expensive. The majority of the complexity originates from the aggregation of negative instances. However, these computationally demanding terms can be transformed with appropriate mathematical manipulations. Here, we take the expanded term for aggregating $\sum_{i\in\mathcal{N}_u^-}\Omega_{ui}^{-(l)}\mathbf{z}_i^{(l)}$ as an example:

$$\begin{aligned}\sum_{i\in\mathcal{N}_u^-}\Omega_{ui}^{-(l)}\mathbf{z}_i^{(l)} &= \sum_{i\in\mathcal{N}_u^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} - b_u^{+(l)} - \alpha}{m-d_u}\mathbf{z}_i^{(l)}\\ &=\left(\sum_{i\in\mathcal{N}_u^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}}{m-d_u}\right) - \frac{(b_u^{+(l)}+\alpha)}{m-d_u}\left(\sum_{i\in\mathcal{N}_u^-}\mathbf{z}_i^{(l)}\right)\end{aligned} \tag{9}$$

and the term $\sum_{i\in\mathcal{N}_u^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}}{m-d_u}$ can be also fast calculated with:

$$\begin{aligned}&\sum_{i\in\mathcal{N}_u^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}}{m-d_u}\\ =&\left(\sum_{i\in\mathcal{I}}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}}{m-d_u}\right)-\left(\sum_{i\in\mathcal{N}_u^+}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}}{m-d_u}\right)\\ =&\frac{(\mathbf{z}_u^{(l)})^T}{m-d_u}\left(\sum_{i\in\mathcal{I}}\mathbf{z}_i^{(l)}(\mathbf{z}_i^{(l)})^T\right)-\left(\sum_{u\in\mathcal{N}_i^+}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}\right)\end{aligned} \tag{10}$$

The complexity of calculating this term can be reduced to $O\big((n+m)d^2+Ed\big)$, where $E$ is the number of edges. Similar methods can be applied to calculating $\Omega_{ui}^{+(l)}, \Omega_{ui}^{-(l)}, C_u^{(l)}, C_i^{(l)}$ with complexity $O\big((n+m)d+Ed\big)$, and calculating $\mathbf{z}_u^{(l+1)}, \mathbf{z}_i^{(l+1)}$ with complexity $O\big((n+m)d^2+Ed\big)$. Readers may refer to the Appendix A.3 for more detailed derivations and complexity analyses. With such an algorithm, although Rankformer involves propagation between every user-item pair, the complexity of Rankformer can be reduced to $O\big((n+m)d^2+Ed\big)$, making it highly efficient and applicable.

We also introduce a few simple strategies during implementation to further enhance training stability: 1) During the calculation of attention weights, we employ embedding normalization. This procedure constrains the value of similarity within a fixed range of [-1,1]. 2) Given that initial embeddings may not be of high quality and could potentially affect the calculation of attention, we employ a warm-up strategy. Specifically, we employ uniform weights and remove negative aggregation at the first layer of Rankformer.

**Table 2: Statistics of datasets.**

| Dataset | #User | #Item | #Interaction |
|---|---|---|---|
| Ali-Display | 17,730 | 10,036 | 173,111 |
| Epinions | 17,893 | 17,659 | 301,378 |
| Amazon-CDs | 51,266 | 46,463 | 731,734 |
| Yelp2018 | 167,037 | 79,471 | 1,970,721 |

## 4 Experiments

In this section, we conduct comprehensive experiments to answer the following research questions:

- **RQ1:** How does Rankformer perform compared to existing state-of-the-art methods?
- **RQ2:** What are the impacts of the important components (such as the terms in Eq(6)) on Rankformer?
- **RQ3:** How do the hyperparameters affect the performance of Rankformer?

## 4.1 Experimental Settings

*4.1.1 Datasets.* We conducted experiments on four conventional real-world datasets: **Ali-Display** [1] provided by Alibaba, is a dataset for estimating click-through rates of Taobao display ads; **Epinions** [49] is a dataset collected from Epinions.com, a popular online consumer review website; **Amazon-CDs** [39] consist of user ratings on products on the Amazon platform; **Yelp2018** [2] is a dataset of user reviews collected by Yelp. We adopt a standard 5-core setting and randomly split the datasets into training, validation, and test sets in a 7:1:2 ratio. The statistical information of the datasets is presented in Table 2.

*4.1.2 Metrics.* We closely refer to [26, 68] and employed two widely used metrics, *Recall@K* and *NDCG@K*, to evaluate the recommendation accuracy. We also simply set $K = 20$ as recent work [68].

*4.1.3 Baselines.* **1) Recommendation Methods.** The following five representative or SOTA recommendation methods are included:

- **MF** [30]: the method exclusively employs the BPR loss function for matrix factorization, without any encoding architecture.
- **LightGCN** [26]: the classical recommendation method that adopts linear GNN.

[1]https://tianchi.aliyun.com/dataset/dataDetail?dataId=56
[2]https://www.yelp.com/dataset

- **DualVAE** [24]: a collaborative recommendation method that combines disentangled representation learning with variational inference.
- **MGFormer** [6]: the state-of-the-art recommendation method based on the graph transformer, equipped with a masking mechanism designed for large-scale graphs.

**2) Graph Transformer with Global Attention Mechanism.** The following three state-of-the-art graph transformer methods are included. We augment these methods with BPR loss to apply them in recommendation tasks:

- **Nodeformer** [62]: the classical linear transformer for large-scale graphs with a kernelized Gumbel-Softmax operator to reduce the algorithmic complexity.
- **DIFFormer** [22]: the linear transformer derived from an energy-constrained diffusion model, applicable to large-scale graphs. This work is built upon NodeFormer.
- **SGFormer** [63]: the latest linear Transformer designed for large-scale graphs, built upon NodeFormer and DIFFormer.

**3) Recommendation Methods With Contrastive Learning Loss.** Given the SOTA methods are often achieved with contrastive learning, we also compare Rankformer with these methods. Nevertheless, it would be unfair as our Rankformer does not utilize constrastive loss. Thus, we also equipped Rankformer with layer-wise constrastive loss as XSimGCL [68] (named as Rankformer-CL). The following baselines are adopted:

- **XSimGCL** [68]: the state-of-the-art method that enhance LightGCN with contrastive learning.
- **GFormer** [32]: the state-of-the-art recommendation method that combine transformer architecture with contrastive learning.

## 4.2 Performance Comparison (RQ1)

The performance comparison of our Rankformer and all baselines in terms of $Recall@20$ and $NDCG@20$ is shown in Table 3. Overall, Rankformer consistently outperforms all comparison methods across all datasets with an average improvement of 4.48%. This result demonstrates the effectiveness of leveraging ranking signals in model architecture.

**Comparison with Recommendation Methods.** Overall, our Rankformer outperforms existing state-of-the-art recommendation methods. Among these comparative methods, graph-based recommender methods such as LightGCN and MGFormer outperform other methods, indicating the advantage of using graph structure in the recommendation. GFormer, which incorporates transformer architecture, outperforms LightGCN using traditional graph GCNs, suggesting that transformer architectures can better leverage collaborative information.

**Comparison with Graph Transformer Methods.** Rankformer consistently outperforms all transformer-based graph representation methods across all datasets. Remarkably, these baseline performances often fall below standard benchmarks and even fail to converge on some datasets, indicating their unsuitability for recommendation tasks. There are two key factors contributing to this outcome: 1) These methods are designed for traditional graph tasks such as node classification, hence they are also based on the smoothness assumption, which does not align with the ranking objectives of recommendation. 2) These methods typically employ a large number of parameters and non-linear modules, making it challenging to effectively train them in RS due to data sparsity.

**Comparison with Graph-based RS with contrastive learning.** The performance of the methods with contrastive learning surpasses other baselines significantly, and adding the contrastive learning loss to Rankformer further enhances recommendation performance noticeably. The Rankformer augmented with the CL-loss notably outperforms these CL-based methods. These observations indicate that contrastive learning can effectively leverage rich collaborative information, and our Rankformer model integrates well with contrastive learning modules.

## 4.3 Ablation Study (RQ2)

To investigate the effects of each module in Rankformer, we conduct an ablation study and the results are presented in Table 4. We draw the following observations:

By removing information aggregation between negative pairs, we observe significant performance drops. This is coincident with our intuition, as the negative relations also bring valuable signals to learn user or item representations.

By removing benchmark terms $b_u$, we also observe performance drops. This term converts absolute scores into relative scores, introducing ranking information.

Removing the offset term $\alpha$ also resulted in a decrease in performance. This term controls the linear term in the Taylor expansion. Table 4 only displays $\alpha = 0$ as an ablation study; specific parameter experiments of $\alpha$ can be found in Figure 3.

By removing the normalization terms $C$ in Rankformer, we observe the terrible performance of Rankformer. It would be trained unstable and usually suffers from gradient explosion. Thus, we introduce normalization terms in the Rankformer, as other transformer models do.

## 4.4 Role of the parameters (RQ3)

**Hyperparameter $\tau$ and the number of Rankformer layers $L$.** The parameter $\tau$ is proportional to the step size used in simulating gradient ascent within Rankformer layers. As illustrated in Figure 2, the performance of Rankformer varies with $\tau$ as the number of layers changes, showing an initial increase followed by a decrease. This trend arises because, with a fixed number of layers, an excessively small step size can result in incomplete optimization, while a step size that is too large may lead to missing the optimal solution. Furthermore, as the number of Rankformer layers $L$ increases, the optimal value of $\tau$ decreases. This is because each Rankformer layer corresponds to one step of gradient ascent. Therefore, with a higher number of steps, approaching the optimal solution gradually with a smaller step size is more effective; whereas with fewer steps, a larger step size is required to converge faster towards the optimal solution.

**Hyperparameter $\alpha$.** The parameter $\alpha$ controls the coefficient of the linear term in the Taylor expansion. Different values of $\alpha$ correspond to different activation functions $\delta(\cdot)$. As shown in Figure 3, when $\alpha$ increases, the performance of Rankformer generally shows an initial improvement followed by a decline. This is because a larger $\alpha$ can better distinguish between positive and negative pairs,

**Table 3: Performance comparison between baselines and Rankformer. The best result is bolded and the runner-up is underlined. The mark '*' suggests the improvement is statistically significant with $p < 0.05$.**

| | | Ali-Display | | Epinions | | Amazon-CDs | | Yelp2018 | |
|---|---|---|---|---|---|---|---|---|---|
| | | ndcg@20 | recall@20 | ndcg@20 | recall@20 | ndcg@20 | recall@20 | ndcg@20 | recall@20 |
| Recommendation Methods | MF | 0.0586 | 0.1094 | 0.0512 | 0.0847 | 0.0717 | 0.1255 | 0.0402 | 0.0766 |
| | LightGCN | 0.0643 | 0.1174 | 0.0523 | 0.0850 | 0.0764 | 0.1317 | 0.0456 | 0.0842 |
| | DualVAE | 0.0603 | 0.1119 | 0.0501 | 0.0822 | 0.0733 | 0.1276 | 0.0402 | 0.0768 |
| | MGFormer | 0.0649 | 0.1083 | 0.0542 | 0.0854 | 0.0763 | 0.1324 | 0.0468 | 0.0869 |
| Graph Transformer | Nodeformer | 0.0205 | 0.0358 | 0.0241 | 0.0446 | - | - | - | - |
| | DIFFormer | 0.0370 | 0.0715 | 0.0269 | 0.0476 | 0.0369 | 0.0681 | 0.0282 | 0.0539 |
| | SGFormer | 0.0411 | 0.0769 | 0.0333 | 0.0571 | 0.0284 | 0.0524 | 0.0218 | 0.0427 |
| Rankformer | | **0.0652** | **0.1208** | **0.0554** | **0.0895** | **0.0831** | **0.1412** | **0.0482** | **0.0890** |
| | | 0.42% | 2.91% | 2.17% | 4.77% | 8.73% | 6.64% | 2.90% | 2.46% |
| Graph-based RS with CL | XSimGCL | 0.0650 | 0.1194 | 0.0558 | 0.0887 | 0.0796 | 0.1346 | 0.0500 | 0.0907 |
| | Gformer | 0.0644 | 0.1176 | 0.0602 | 0.0978 | 0.0812 | 0.1366 | 0.0502 | 0.0897 |
| Rankformer - CL | | **0.0680** | **0.1246** | **0.0633** | **0.1008** | **0.0877** | **0.1467** | **0.0523** | **0.0941** |
| | | 4.57% | 4.32% | 5.22% | 3.11% | 8.00% | 7.41% | 4.29% | 3.71% |

**Table 4: The results of the ablation study, where negative pairs, benchmark terms and normalization terms have been removed, respectively.**

| | Ali-Display | | Epinions | | Amazon-CDs | | Yelp2018 | |
|---|---|---|---|---|---|---|---|---|
| | ndcg@20 | recall@20 | ndcg@20 | recall@20 | ndcg@20 | recall@20 | ndcg@20 | recall@20 |
| Rankformer - w/o negative pairs | 0.0535 | 0.1026 | 0.0409 | 0.0696 | 0.0744 | 0.1295 | 0.0429 | 0.0800 |
| Rankformer - w/o benchmark | 0.0641 | 0.1189 | 0.0535 | 0.0875 | 0.0827 | 0.1406 | 0.0475 | 0.0884 |
| Rankformer - w/o offset | 0.0539 | 0.1034 | 0.0544 | 0.0887 | 0.0732 | 0.1268 | 0.0349 | 0.0665 |
| Rankformer - w/o normalization of Ω | 0.0291 | 0.0527 | 0.0351 | 0.0617 | 0.0357 | 0.0596 | 0.0166 | 0.0321 |
| Rankformer | **0.0652** | **0.1208** | **0.0554** | **0.0895** | **0.0831** | **0.1412** | **0.0482** | **0.0890** |

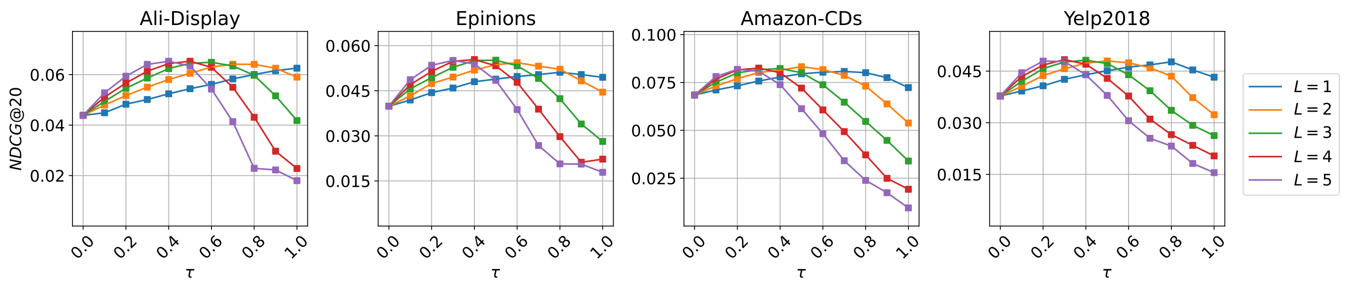

**Figure 2: Performance of Rankformer in terms of $NDCG@20$ with different layers $L$ and hyperparameter $\tau$.**

but an excessively large $\alpha$ can lead to over-smoothing of the representations learned by Rankformer. When $\alpha$ approaches positive infinity, the model will degenerate into a GCN that aggregates only the average representation of the entire graph and the representation of node neighborhoods.

## 5 Related Work

### 5.1 Architectures of Recommendation Models

In recent years, there has been a surge of publications on recommendation model architectures, ranging from traditional matrix factorization [28, 30, 40], neighbor-based methods [54], to more advanced auto-encoders [15, 34, 66], diffusion models [61], large language models [35, 53], knowledge distillation [13, 27], offline reinforcement learning [19, 20], recurrent neural networks [12, 23], graph neural networks [4, 16, 26, 52], and Transformer-based methods. In this section, we focus primarily on reviewing the most relevant GNN-based and Transformer-based recommendation models and refer readers to excellent surveys for more comprehensive information [1, 21, 46, 70].

Graph Neural Networks (GNNs), which leverage message-passing mechanisms to fully exploit collaborative information within graphs,

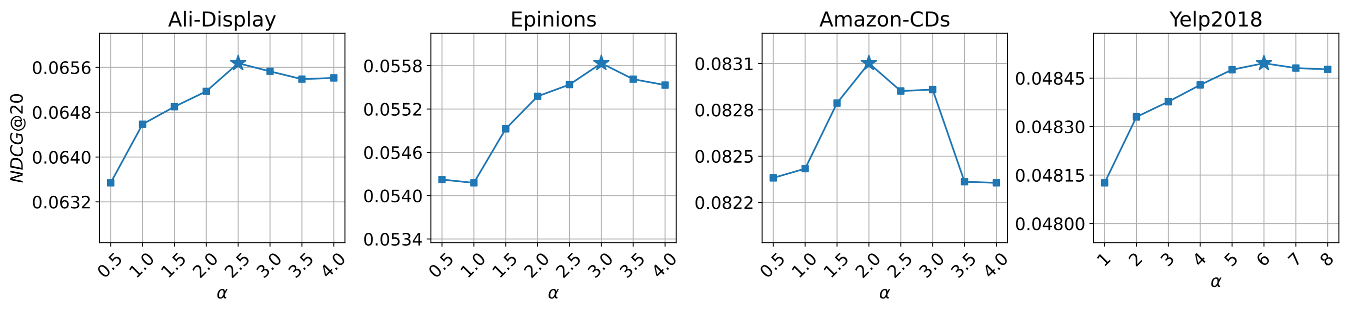

**Figure 3: Performance in terms of $NDCG@20$ with different hyperparameter $\alpha$.**

have demonstrated remarkable effectiveness in the recommendation systems (RS) domain in recent years. Early work [18, 25, 55, 67] directly inherits the GNN architecture, including complex parameters, from the graph learning domain. Later, the representative LightGCN [26] pruned unnecessary parameters from GNNs, yielding better and more efficient performance. Building on LightGCN, recent work has made various improvements. For instance, some researchers [2, 59, 68, 69] have explored the use of contrastive learning strategies[57] in GNN-based methods, achieving state-of-the-art performance; others [56] have refined and reweighted graph structures to better suit the recommendation task. Additionally, some works [43] have analyzed and enhanced LightGCN from a spectral perspective. While these GNN-based methods have achieved significant success, a potential limitation is that the role of GNNs tends to deviate from the ranking objective, which may hinder their effectiveness.

Regarding the Transformer architecture, it was initially introduced to sequential recommendation as a superior alternative to recurrent neural networks [48] for modeling item dependencies in sequences. Additionally, Transformers have been employed in multimodal recommendation tasks to better fuse multimodal information [36]. However, these Transformer architectures differ significantly from our proposed Rankformer, as they are not tailored for generic recommendation scenarios and do not leverage global aggregation across all users and items. To the best of our knowledge, there are three related works that explore generic recommendation: GFormer [32], SIGformer [11], and MGFormer [6]. *We argue that the major limitations of these methods are heuristic designs that do not incorporate the ranking property into the architecture.* Moreover, they exhibit additional limitations: 1) SIGformer requires signed interaction information, which may not always be available; 2) GFormer utilizes the Transformer for generating contrastive views, rather than as the recommendation backbone; 3) MGFormer involves complex masking operations and positional encoding, making it difficult to train effectively and less efficient.

### 5.2 Graph Transformer

In recent years, there has been an increasing number of works applying Transformer to graph learning. By utilizing its global attention mechanism, Transformer enables each node on the graph to aggregate information from all nodes in the graph at each step, mitigating issues such as over-smoothing, over-squeezing, and expressive boundary problems caused by traditional GNNs that only aggregate low-order neighbors [64]. Research on graph Transformers mainly focused on designing positional encodings for capturing topological structures, including spectral encoding [17, 31], centrality encoding [65], shortest path encoding [33, 65], and substructure encoding [3], which are suitable for positional encoding in graphs.

Given the time and space complexities of Transformers with global pairwise attention are usually proportional to the square of the number of nodes, the early study on graph transformer can only limited to small graphs. To tackle this, various acceleration strategies have been developed. NodeFormer [62] replaces the original attention matrix computation with a positive definite kernel to achieve linear computational complexity. Methods like Gophormer [73], ANS-GT [72], and NAGphormer [9] reduce computational complexity through sampling strategies, while DIFFormer [61] derives a linearizable Transformer using a diffusion model.

## 6 Conclusions

The personalized ranking is a fundamental attribute of Recommender Systems (RS). In this work, we propose Rankformer, which explicitly incorporates this crucial property into its architectural design. Rankformer simulates the gradient descent process of the ranking objective and introduces a unique Transformer architecture. This specific design facilitates the evolution of embeddings in a direction that enhances ranking performance. The current Rankformer is based on the BPR loss, but its design principles may be applicable to a wider range of more advanced loss functions, such as enhanced BPR loss [10] and softmax loss[58, 60]. In the future, it would be of great interest to extend this architecture to other recommendation scenarios, such as sequential recommendation and LLM-based recommendation, allowing ranking signals to be seamlessly integrated into the architectures.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (62476244,62372399,62476245), Ant Group Research Intern Program, Zhejiang Provincial Natural Science Foundation of China (Grant No: LTGG23F030005).

## References

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *Comput. Surveys* 55, 7 (2022), 1–38.

[2] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In *The Eleventh International Conference on Learning Representations*.

[3] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. 2022. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*. PMLR, 3469–3489.

[4] Hao Chen, Yuanchen Bei, Qijie Shen, Yue Xu, Sheng Zhou, Wenbing Huang, Feiran Huang, Senzhang Wang, and Xiao Huang. 2024. Macro graph neural networks for online billion-scale recommender systems. In *Proceedings of the ACM on Web Conference 2024*. 3598–3608.

[5] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. 2021. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12299–12310.

[6] Huiyuan Chen, Zhe Xu, Chin-Chia Michael Yeh, Vivian Lai, Yan Zheng, Minghua Xu, and Hanghang Tong. 2024. Masked graph transformer for large-scale recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2502–2506.

[7] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to debias for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 21–30.

[8] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

[9] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. 2022. NAGphormer: A tokenized graph transformer for node classification in large graphs. In *The Eleventh International Conference on Learning Representations*.

[10] Jiawei Chen, Junkang Wu, Jiancan Wu, Xuezhi Cao, Sheng Zhou, and Xiangnan He. 2023. Adap-$\tau$: Adaptively modulating embedding magnitude for recommendation. In *Proceedings of the ACM Web Conference 2023*. 1085–1096.

[11] Sirui Chen, Jiawei Chen, Sheng Zhou, Bohao Wang, Shen Han, Chanfei Su, Yuqing Yuan, and Can Wang. 2024. SIGformer: Sign-aware Graph Transformer for Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1274–1284.

[12] Yan Chen, Wanhui Qian, Dongqin Liu, Mengdi Zhou, Yipeng Su, Jizhong Han, and Ruixuan Li. 2022. Your Social Circle Affects Your Interests: Social Influence Enhanced Session-Based Recommendation. In *International Conference on Computational Science*. Springer, 549–562.

[13] Yu Cui, Feng Liu, Pengbo Wang, Bohao Wang, Heng Tang, Yi Wan, Jun Wang, and Jiawei Chen. 2024. Distillation matters: empowering sequential recommenders to match the performance of large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 507–517.

[14] José de Jesús Rubio. 2020. Stability analysis of the modified Levenberg–Marquardt algorithm for the artificial neural network training. *IEEE transactions on neural networks and learning systems* 32, 8 (2020), 3510–3524.

[15] Shuiguang Deng, Longtao Huang, Guandong Xu, Xindong Wu, and Zhaohui Wu. 2016. On deep learning for trust-aware recommendations in social networks. *IEEE transactions on neural networks and learning systems* 28, 5 (2016), 1164–1177.

[16] Hande Dong, Jiawei Chen, Fuli Feng, Xiangnan He, Shuxian Bi, Zhaolin Ding, and Peng Cui. 2021. On the equivalence of decoupled graph convolution network and label propagation. In *Proceedings of the Web Conference 2021*. 3651–3662.

[17] Vijay Prakash Dwivedi and Xavier Bresson. 2021. A Generalization of Transformer Networks to Graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications* (2021).

[18] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[19] Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. 2023. Alleviating matthew effect of offline reinforcement learning in interactive recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 238–248.

[20] Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2023. CIRS: Bursting filter bubbles by counterfactual interactive recommender system. *ACM Transactions on Information Systems* 42, 1 (2023), 1–27.

[21] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems* 1, 1 (2023), 1–51.

[22] Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. 2022. Difformer: Empowering diffusion models on the embedding space for text generation. *arXiv preprint arXiv:2212.09412* (2022).

[23] Pan Gu, Yuqiang Han, Wei Gao, Guandong Xu, and Jian Wu. 2021. Enhancing session-based social recommendation through item graph embedding and contextual friendship modeling. *Neurocomputing* 419 (2021), 190–202.

[24] Zhiqiang Guo, Guohui Li, Jianjun Li, Chaoyang Wang, and Si Shi. 2024. DualVAE: Dual Disentangled Variational AutoEncoder for Recommendation. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. SIAM, 571–579.

[25] Zhiwei Guo and Heng Wang. 2020. A deep graph neural network-based mechanism for social recommendations. *IEEE Transactions on Industrial Informatics* 17, 4 (2020), 2776–2783.

[26] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[27] Feiran Huang, Zefan Wang, Xiao Huang, Yufeng Qian, Zhetao Li, and Hao Chen. 2023. Aligning distillation for cold-start item recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1147–1157.

[28] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*. 135–142.

[29] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. 2022. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics* 11, 1 (2022), 141.

[30] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[31] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems* 34 (2021), 21618–21629.

[32] Chaoliu Li, Lianghao Xia, Xubin Ren, Yaowen Ye, Yong Xu, and Chao Huang. 2023. Graph Transformer for Recommendation. *arXiv preprint arXiv:2306.02330* (2023).

[33] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems* 33 (2020), 4465–4478.

[34] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.

[35] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1785–1795.

[36] Yong Liu, Susen Yang, Chenyi Lei, Guoxin Wang, Haihong Tang, Juyong Zhang, Aixin Sun, and Chunyan Miao. 2021. Pre-training graph transformer with multimodal side information for recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*. 2853–2861.

[37] Xin-Long Luo, Li-Zhi Liao, and Hon Wah Tam. 2007. Convergence analysis of the Levenberg–Marquardt method. *Optimization Methods and Software* 22, 4 (2007), 659–678.

[38] Hao Ma, Irwin King, and Michael R Lyu. 2009. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 203–210.

[39] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*. 897–908.

[40] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *Advances in neural information processing systems* 20 (2007).

[41] Jorge J Moré. 2006. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*. Springer, 105–116.

[42] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).

[43] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. Less is more: Reweighting important spectral graph features for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1273–1282.

[44] Ananth Ranganathan. 2004. The levenberg-marquardt algorithm. *Tutoral on LM algorithm* 11, 1 (2004), 101–110.

[45] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[46] Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar. 2024. A survey of graph neural networks for social recommender systems. *Comput. Surveys* 56, 10 (2024), 1–34.

[47] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B Khaled Letaief, and Dongsheng Li. 2021. How powerful is graph convolution for recommendation?. In *Proceedings of the 30th ACM international conference on information & knowledge management*. 1619–1629.

[48] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[49] Jiliang Tang, Huiji Gao, Huan Liu, and Atish Das Sarma. 2012. eTrust: Understanding trust evolution in an online world. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 253–261.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[51] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[52] Bohao Wang, Jiawei Chen, Changdong Li, Sheng Zhou, Qihao Shi, Yang Gao, Yan Feng, Chun Chen, and Can Wang. 2024. Distributionally Robust Graph-based Recommendation System. *arXiv preprint arXiv:2402.12994* (2024).

[53] Bohao Wang, Feng Liu, Jiawei Chen, Yudi Wu, Xingyu Lou, Jun Wang, Yan Feng, Chun Chen, and Can Wang. 2024. Llm4dsr: Leveraing large language model for denoising sequential recommendation. *arXiv preprint arXiv:2408.08208* (2024).

[54] Dawei Wang, Yuehwern Yih, and Mario Ventresca. 2020. Improving neighbor-based collaborative filtering by using a hybrid similarity measurement. *Expert Systems with Applications* 160 (2020), 113651.

[55] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[56] Yu Wang, Yuying Zhao, Yi Zhang, and Tyler Derr. 2023. Collaboration-aware graph convolutional network for recommender systems. In *Proceedings of the ACM Web Conference 2023*. 91–101.

[57] Junkang Wu, Jiawei Chen, Jiancan Wu, Wentao Shi, Xiang Wang, and Xiangnan He. 2024. Understanding contrastive learning via distributionally robust optimization. *Advances in Neural Information Processing Systems* 36 (2024).

[58] Junkang Wu, Jiawei Chen, Jiancan Wu, Wentao Shi, Jizhi Zhang, and Xiang Wang. 2024. Bsl: Understanding and improving softmax loss for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 816–830.

[59] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 726–735.

[60] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, and Tianyu Qiu. 2024. On the effectiveness of sampled softmax loss for item recommendation. *ACM Transactions on Information Systems* 42, 4 (2024), 1–26.

[61] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 235–244.

[62] Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems* 35 (2022), 27387–27401.

[63] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. 2023. Simplifying and Empowering Transformers for Large-Graph Representations. *arXiv preprint arXiv:2306.10759* (2023).

[64] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.

[65] 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? *Advances in Neural Information Processing Systems* 34 (2021), 28877–28888.

[66] Haochao Ying, Liang Chen, Yuwen Xiong, and Jian Wu. 2016. Collaborative deep ranking: A hybrid pair-wise recommendation algorithm with implicit feedback. In *Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part II 20*. Springer, 555–567.

[67] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[68] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2023. XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[69] 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 *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1294–1303.

[70] 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.

[71] Wenhui Yu and Zheng Qin. 2020. Graph convolutional network for recommendation with low-pass collaborative filters. In *International Conference on Machine Learning*. PMLR, 10936–10945.

[72] Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. 2022. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems* 35 (2022), 21171–21183.

[73] 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 preprint arXiv:2110.13094* (2021).

[74] Zihao Zhao, Jiawei Chen, Sheng Zhou, Xiangnan He, Xuezhi Cao, Fuzheng Zhang, and Wei Wu. 2022. Popularity bias is not always evil: Disentangling benign and harmful bias for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 10 (2022), 9920–9931.

## A Appendices

### A.1 Derivation of Rankformer Layer

According to Eq(1), we have:

$$\mathcal{L}(\mathbf{Z};\sigma) = -\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{N}_u^+}\sum_{j\in\mathcal{N}_u^-}\frac{\sigma(\mathbf{z}_u^T\mathbf{z}_i - \mathbf{z}_u^T\mathbf{z}_j)}{d_u(m-d_u)} + \lambda\|\mathbf{Z}\|_2^2 \tag{11}$$

By approximating the function $\sigma(x)$ using a second-order Taylor expansion, we obtain:

$$\widetilde{\mathcal{L}}(\mathbf{Z};\sigma) = -\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{N}_u^+}\sum_{j\in\mathcal{N}_u^-}\frac{1}{d_u(m-d_u)}\left[\omega_{uij}(\mathbf{z}_u^T\mathbf{z}_i - \mathbf{z}_u^T\mathbf{z}_j)\right] + \lambda\|\mathbf{Z}\|_2^2 \tag{12}$$

where $\omega_{uij} = \mathbf{z}_u^T\mathbf{z}_i - \mathbf{z}_u^T\mathbf{z}_j + \alpha$. $\alpha > 0$ is a hyperparameter that controls the coefficient of the linear term in the Taylor expansion of $\sigma(\cdot)$.

The optimization objective is to minimize the function $\mathcal{L}(\mathbf{Z};\sigma)$. Therefore, we propose performing one step gradient descent with the step size of $\tau$ on $\widetilde{\mathcal{L}}(\mathbf{Z};\sigma)$ within each Rankformer layer:

$$\mathbf{Z}^{(l+1)} = \mathbf{Z}^{(l)} - \tau\cdot\frac{\partial}{\partial\mathbf{Z}^{(l)}}\widetilde{\mathcal{L}}(\mathbf{Z}^{(l)};\sigma) \tag{13}$$

$$= \mathbf{Z}^{(l)} + \tau\cdot(\mathbf{\Omega}^{(l)}\mathbf{Z} - \lambda\mathbf{Z}^{(l)}) \tag{14}$$

$$= (1-\tau\lambda)\mathbf{Z}^{(l)} + \tau\cdot\mathbf{\Omega}^{(l)}\mathbf{Z}^{(l)} \tag{15}$$

where $\mathbf{\Omega}_{ab}^{(l)} = \begin{cases}\Omega_{ab}^{+\ (l)} & , b\in\mathcal{N}_a^+ \\ \Omega_{ab}^{-\ (l)} & , b\in\mathcal{N}_a^- \\ 0 & , \text{otherwise}\end{cases}$ is:

$$\Omega_{ui}^{+\ (l)} = \Omega_{iu}^{+\ (l)} = \begin{cases}\sum_{j\in\mathcal{N}_u^-}\frac{\omega_{uij}}{d_u(m-d_u)} & , u\in\mathcal{U}, i\in\mathcal{N}_u^+ \\ 0 & , u\in\mathcal{U}, i\in\mathcal{N}_u^-\end{cases} \tag{16}$$

$$\Omega_{ui}^{-\ (l)} = \Omega_{iu}^{-\ (l)} = \begin{cases}0 & , u\in\mathcal{U}, i\in\mathcal{N}_u^+ \\ -\sum_{j\in\mathcal{N}_u^+}\frac{\omega_{uji}}{d_u(m-d_u)} & , u\in\mathcal{U}, i\in\mathcal{N}_u^-\end{cases} \tag{17}$$

We simply set the hyperparameter $\lambda = 1$, so we get

$$\mathbf{Z}^{(l+1)} = (1-\tau)\mathbf{Z}^{(l)} + \tau\mathbf{\Omega}^{(l)}\mathbf{Z}^{(l)} \tag{18}$$

By reorganization the terms, we can easily get the Eq.(5).

### A.2 The proof of Lemma 1

Proof. Let $\mathcal{L}(\theta)$ be the loss function of a model (*i.e.,* BPR), and $\theta$ be the parameters of the model. We assume $\mathcal{L}(\theta)$ has bounded gradient and Hessian matrix, i.e., $|\mathcal{L}(\theta)| \le C$ and $|\mathbf{H}|_2 \le C$. These assumptions are mild and standard for convergence and gradient analyses.

A one-layer Rankformer can be formulated as:

$$\mathcal{L}_R(\theta) = \mathcal{L}(\theta - \tau\nabla\mathcal{L}(\theta)) \tag{19}$$

where $\tau$ denotes the step-size, and $\theta - \tau\nabla\mathcal{L}(\theta)$ denotes the gradient descent that Rankformer simulates.

The gradient of $\mathcal{L}_R(\theta)$ *w.r.t.* $\theta$ can be written as:

$$\nabla\mathcal{L}_R(\theta) = (I - \tau\mathbf{H})\nabla\mathcal{L}(\theta - \tau\nabla\mathcal{L}(\theta)) \tag{20}$$

where we apply the chain rule, and $\mathbf{H}$ denotes the Hessian matrix of $L(\theta)$.

By performing a Taylor expansion of $\nabla\mathcal{L}(\theta - \tau\nabla\mathcal{L}(\theta))$, we have:

$$\nabla\mathcal{L}_R(\theta) = (I - \tau\mathbf{H})\nabla(\mathcal{L}(\theta) - \tau(\nabla\mathcal{L}(\theta))^T\nabla\mathcal{L}(\theta)) + O((\tau C)^2) \tag{21}$$

$$= (I - \tau\mathbf{H})(\nabla\mathcal{L}(\theta) - 2\tau\mathbf{H}\nabla\mathcal{L}(\theta)) + O((\tau C)^2) \tag{22}$$

$$= (I - 3\tau\mathbf{H})\nabla\mathcal{L}(\theta) + O((\tau C)^2) \tag{23}$$

$$= (I + 3\tau\mathbf{H})^{-1}\nabla\mathcal{L}(\theta) + O((\tau C)^2) \tag{24}$$

The last equation holds due to the binomial expansion of $(I + 3\tau\mathbf{H})^{-1}$, omitting higher-order terms of $\tau$. This formula reveals the theoretical advantage of the Rankformer architecture, which refines the original gradient by multiplying it with the matrix $(I + 3\tau\mathbf{H})^{-1}$. In fact, utilizing the gradient $(I + 3\tau\mathbf{H})^{-1}\nabla\mathcal{L}(\theta)$ for optimization is known as Levenberg–Marquardt algorithm [44], which interpolates between gradient descent ($\nabla\mathcal{L}(\theta)$) and the Newton optimization method ($\mathbf{H}^{-1}\nabla\mathcal{L}(\theta)$). The introduction of the Hessian matrix (second-order derivative information) provides more accurate optimization steps towards the minimum. This algorithm demonstrates faster convergence (i.e., quadratic convergence), stability, and improved performance [14, 37, 41]. However, a significant challenge of the Levenberg–Marquardt algorithm is that it requires explicit calculation of the complex Hessian matrix, which is time-consuming, making this algorithm less applicable in deep neural networks. Remarkably, our Rankformer can approximate this effective algorithm, benefiting from its advantages without requiring explicit calculation of the complex Hessian matrix. □

## A.3 Details of Fast Implementation

Eq(5) - Eq(8) are computed as follows, achieving a complexity of $O\big((n+m)d^2 + Ed\big)$, where $E$ is the number of edges.

**Step 1:** Calculate $b_u^{+(l)}$ and $b_u^{-(l)}$ with a complexity of $O\big((n+m)d + Ed\big)$.

$$b_u^{+(l)} = \frac{1}{d_u}\sum_{i\in\mathcal{N}_u^+}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} \tag{25}$$

$$= \frac{1}{d_u}(\mathbf{z}_u^{(l)})^T\left(\sum_{i\in\mathcal{N}_u^+}\mathbf{z}_i^{(l)}\right) \tag{26}$$

$$b_u^{-(l)} = \frac{1}{m-d_u}\sum_{i\in\mathcal{N}_u^-}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} \tag{27}$$

$$= \frac{1}{m-d_u}(\mathbf{z}_u^{(l)})^T\left(\sum_{i\in\mathcal{N}_u^-}\mathbf{z}_i^{(l)}\right) \tag{28}$$

$$= \frac{1}{m-d_u}(\mathbf{z}_u^{(l)})^T\left(\sum_{i\in\mathcal{I}}\mathbf{z}_i^{(l)} - \sum_{i\in\mathcal{N}_u^+}\mathbf{z}_i^{(l)}\right) \tag{29}$$

Similarly, all terms resembling $\sum_{i\in\mathcal{N}_u^-}\mathbf{x}_i$ can be transformed into $\sum_{i\in\mathcal{I}}\mathbf{x}_i - \sum_{i\in\mathcal{N}_u^+}\mathbf{x}_i$ and computed linearly, and all terms resembling $\sum_{u\in\mathcal{N}_i^-}\mathbf{x}_u$ can be transformed into $\sum_{u\in\mathcal{U}}\mathbf{x}_u - \sum_{u\in\mathcal{N}_i^+}\mathbf{x}_u$ and computed linearly. Therefore, $\sum_{i\in\mathcal{N}_u^-}$ and $\sum_{u\in\mathcal{N}_i^-}$ will be retained without expansion in the subsequent derivation.

**Step 2:** Calculate $C_u{}^{(l)}$, $C_i{}^{(l)}$ with a complexity of $O\big((n+m)d + Ed\big)$.

Since we have normalized $\mathbf{z}$, $(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} \le 1$. When $\alpha \ge 2$, $\Omega_{ui}^{+}{}^{(l)} = \Omega_{iu}^{+}{}^{(l)} > 0$, and $\Omega_{ui}^{+}{}^{(l)} = \Omega_{iu}^{+}{}^{(l)} < 0$, so we can remove the absolute value symbols.

$$\sum_{i\in\mathcal{N}_u^+}\left|\Omega_{ui}^{+}{}^{(l)}\right| = \sum_{i\in\mathcal{N}_u^+}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} - b_u^{-(l)} + \alpha}{d_u} \tag{30}$$

$$= \frac{(\mathbf{z}_u^{(l)})^T}{d_u}\left(\sum_{i\in\mathcal{N}_u^+}\mathbf{z}_i^{(l)}\right) - b_u^{-(l)} + \alpha \tag{31}$$

$$= b_u^{+(l)} - b_u^{-(l)} + \alpha \tag{32}$$

$$\sum_{i\in\mathcal{N}_u^-}\left|\Omega_{ui}^{-}{}^{(l)}\right| = -\sum_{i\in\mathcal{N}_u^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} - b_u^{+(l)} - \alpha}{m-d_u} \tag{33}$$

$$= -\frac{(\mathbf{z}_u^{(l)})^T}{m-d_u}\left(\sum_{i\in\mathcal{N}_u^-}\mathbf{z}_i^{(l)}\right) + b_u^{+(l)} + \alpha \tag{34}$$

$$= -b_u^{-(l)} + b_u^{+(l)} + \alpha \tag{35}$$

$$\sum_{u\in\mathcal{N}_i^+}\left|\Omega_{iu}^{+}{}^{(l)}\right| = \sum_{u\in\mathcal{N}_i^+}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} - b_u^{-(l)} + \alpha}{d_u} \tag{36}$$

$$= (\mathbf{z}_i^{(l)})^T\left(\sum_{u\in\mathcal{N}_i^+}\frac{\mathbf{z}_u^{(l)}}{d_u}\right) - \left(\sum_{u\in\mathcal{N}_i^+}\frac{b_u^{-(l)}}{d_u}\right) + \alpha\left(\sum_{u\in\mathcal{N}_i^+}\frac{1}{d_u}\right) \tag{37}$$

$$\sum_{u\in\mathcal{N}_i^-}\left|\Omega_{iu}^{-}{}^{(l)}\right| = -\sum_{u\in\mathcal{N}_i^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} - b_u^{+(l)} - \alpha}{m-d_u} \tag{38}$$

$$= -(\mathbf{z}_i^{(l)})^T\left(\sum_{u\in\mathcal{N}_i^-}\frac{\mathbf{z}_u^{(l)}}{m-d_u}\right) + \left(\sum_{u\in\mathcal{N}_i^-}\frac{b_u^{+(l)}}{m-d_u}\right) + \alpha\left(\sum_{u\in\mathcal{N}_i^-}\frac{1}{m-d_u}\right) \tag{39}$$

Combining the above formulas, we get $C_u^{(l)}$ and $C_i^{(l)}$ with a complexity of $O\big((n+m)d + Ed\big)$.

**Step 3:** Calculate $\sum\limits_{i\in\mathcal{N}_u^+}\Omega_{ui}^{+}{}^{(l)}\mathbf{z}_i^{(l)}$, $\sum\limits_{i\in\mathcal{N}_u^-}\Omega_{ui}^{-}{}^{(l)}\mathbf{z}_i^{(l)}$, $\sum\limits_{u\in\mathcal{N}_i^+}\Omega_{iu}^{+}{}^{(l)}\mathbf{z}_u^{(l)}$ and $\sum\limits_{u\in\mathcal{N}_i^-}\Omega_{iu}^{-}{}^{(l)}\mathbf{z}_u^{(l)}$ with a complexity of $O\big((n+m)d^2 + Ed\big)$.

$$\sum_{i\in\mathcal{N}_u^+}\Omega_{ui}^{+}{}^{(l)}\mathbf{z}_i^{(l)} = \sum_{i\in\mathcal{N}_u^+}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} - b_u^{-(l)} + \alpha}{d_u}\mathbf{z}_i^{(l)} \tag{40}$$

$$= \frac{1}{d_u}\left(\sum_{i\in\mathcal{N}_u^+}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}\right) - \frac{b_u^{-(l)} - \alpha}{d_u}\left(\sum_{i\in\mathcal{N}_u^+}\mathbf{z}_i^{(l)}\right) \tag{41}$$

$$\sum_{i\in\mathcal{N}_u^-}\Omega_{ui}^{-}{}^{(l)}\mathbf{z}_i^{(l)} = \sum_{i\in\mathcal{N}_u^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)} - b_u^{+(l)} - \alpha}{m-d_u}\mathbf{z}_i^{(l)} \tag{42}$$

**Table 5: Runtime comparison of Rankformer with baselines (in seconds).**

| | Ali-Display | Epinions | Amazon-CDs | Yelp2018 |
|---|---|---|---|---|
| LightGCN (source code) | 1687 | 2773 | 11694 | 21651 |
| LightGCN (rewrite) | 99 | 62 | 91 | 1132 |
| MGFormer | 1192 | 1580 | 3090 | 2435 |
| GFormer | 2250 | 3523 | 6779 | 36587 |
| Rankformer | 109 | 149 | 282 | 2212 |

$$=\frac{1}{m-d_u}\left(\sum_{i\in\mathcal{N}_u^-}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}\right)-\frac{b_u^{+(l)}+\alpha}{m-d_u}\left(\sum_{i\in\mathcal{N}_u^-}\mathbf{z}_i^{(l)}\right) \tag{43}$$

$$\sum_{u\in\mathcal{N}_i^+}\Omega_{iu}^{+\,(l)}\mathbf{z}_u^{(l)}=\sum_{u\in\mathcal{N}_i^+}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}-b_u^{-(l)}+\alpha}{d_u}\mathbf{z}_u^{(l)} \tag{44}$$

$$=\left(\sum_{u\in\mathcal{N}_i^+}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_u^{(l)}}{d_u}\right)-\left(\sum_{u\in\mathcal{N}_i^+}\frac{(b_u^{-(l)}-\alpha)\mathbf{z}_u^{(l)}}{d_u}\right) \tag{45}$$

$$\sum_{u\in\mathcal{N}_i^-}\Omega_{iu}^{-\,(l)}\mathbf{z}_u^{(l)}=\sum_{u\in\mathcal{N}_i^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}-b_u^{+(l)}-\alpha}{m-d_u}\mathbf{z}_u^{(l)} \tag{46}$$

$$=\left(\sum_{u\in\mathcal{N}_i^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_u^{(l)}}{m-d_u}\right)-\left(\sum_{u\in\mathcal{N}_i^-}\frac{(b_u^{+(l)}+\alpha)\mathbf{z}_u^{(l)}}{m-d_u}\right) \tag{47}$$

where $\sum_{i\in\mathcal{N}_u^-}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}$ and $\sum_{u\in\mathcal{N}_i^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_u^{(l)}}{m-d_u}$ can be calculated as follows with a complexity of $O\big((n+m)d^2+Ed\big)$:

$$\sum_{i\in\mathcal{N}_u^-}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}$$

$$=\left(\sum_{i\in\mathcal{I}}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}\right)-\left(\sum_{i\in\mathcal{N}_u^+}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}\right) \tag{48}$$

$$=(\mathbf{z}_u^{(l)})^T\left(\sum_{i\in\mathcal{I}}\mathbf{z}_i^{(l)}(\mathbf{z}_i^{(l)})^T\right)-\left(\sum_{i\in\mathcal{N}_u^+}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}\right) \tag{49}$$

$$\sum_{u\in\mathcal{N}_i^-}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_u^{(l)}}{m-d_u}$$

$$=\left(\sum_{u\in\mathcal{U}}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_u^{(l)}}{m-d_u}\right)-\left(\sum_{u\in\mathcal{N}_i^+}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_u^{(l)}}{m-d_u}\right) \tag{50}$$

$$=(\mathbf{z}_i^{(l)})^T\left(\sum_{u\in\mathcal{U}}\frac{\mathbf{z}_u^{(l)}(\mathbf{z}_u^{(l)})^T}{m-d_u}\right)-\left(\sum_{u\in\mathcal{N}_i^+}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_u^{(l)}}{m-d_u}\right) \tag{51}$$

The time and space complexity of calculating $\sum_{i\in\mathcal{N}_u^+}(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_i^{(l)}$ and $\sum_{u\in\mathcal{N}_i^+}\frac{(\mathbf{z}_u^{(l)})^T\mathbf{z}_i^{(l)}\mathbf{z}_u^{(l)}}{m-d_u}$ is $O(Ed)$, while the time and space complexity of calculating $(\mathbf{z}_u^{(l)})^T\left(\sum_{i\in\mathcal{I}}\mathbf{z}_i^{(l)}(\mathbf{z}_i^{(l)})^T\right)$ and $(\mathbf{z}_i^{(l)})^T\left(\sum_{u\in\mathcal{U}}\frac{\mathbf{z}_u^{(l)}(\mathbf{z}_u^{(l)})^T}{m-d_u}\right)$ is $O\big((n+m)d^2\big)$.

Therefore, the overall time and space complexity is $O\big((n+m)d^2+Ed\big)$.

## A.4 Experimental Parameter Settings

For Rankformer, we adopt the Adam optimizer and search the hyperparameters with grid search. Specifically, we set the hidden embedding dimension $d = 64$ as recent works [26]. The weight decay is set to $1e-4$. We search for $\tau$ with a step size of 0.1 within the range $[0,1]$, and select the number of layers $L$ for Rankformer in the range of $\{1,2,3,4,5\}$. To reduce the number of hyperparameters, except for the experiments in Figure 3, the parameter $\alpha$ is simply set to 2, although Figure 3 indicates that fine-tuning this parameter could improve the model's performance. For Rankformer without contrastive learning loss, the learning rate is set to 0.1.

We also test Rankformer-CL which combines Rankformer with the layer-wise contrastive loss used in XSimGCL. For Rankformer-CL, the batch size is set to 2048, and the learning rate is set to 0.001. The relevant parameters for the contrastive loss are simply set as: $\epsilon_{\text{cl}} = 0.2$, $\lambda_{\text{cl}} = 0.05$, $\tau_{\text{cl}} = 0.15$.

For the compared methods, we use the source code provided officially and follow the instructions from the original paper to search for the optimal hyperparameters. We extensively traversed and expanded the entire hyperparameter space as recommended by the authors to ensure that all compared methods achieved optimal performance.

## A.5 Efficiency Comparison

Table 5 illustrates the actual runtime comparison between Rankformer and other baselines on four datasets. The official implementation of LightGCN exhibits low efficiency, prompting us to introduce a re-implemented version of LightGCN for comparison. This version aligns with our Rankformer source code in data processing, training, testing, and other aspects, differing only in the encoding architecture. The experiments demonstrate that our Rankformer, with a complexity of $O((n+m)d^2+Ed)$, exhibits similar actual runtime to LightGCN, which has a complexity of $O(Ed)$. In comparison to other recommendation methods with graph transformer such as MGFormer and GFormer, Rankformer exhibits significantly faster runtime efficiency.