electronics

MDPI

Article

Explicitly Exploiting Implicit User and Item Relations in Graph Convolutional Network (GCN) for Recommendation

Bowen Xiao * and Deng Chen

Department of Computer Science and Engineering, Wuhan Institute of Technology, 19 Liufang Avenue, Jiangxia District, Wuhan 430079, China; dchen@wit.edu.cn

* Correspondence: bw.xiao.2004@gmail.com

Abstract: Most existing collaborative filtering-based recommender systems rely solely on available user–item interactions for user and item representation learning. Their performance often suffers significantly when interactions are sparse, as limited user and item interactions are insufficient for learning robust representations. To address this issue, recent research has explored additional information between users and items by leveraging the user–item bipartite graph. However, these methods have not fully exploited high-order neighborhood information, primarily using sampled interactions to enrich training data rather than integrating this information directly into representation learning. In this paper, we propose a novel model, EIR-GCN (Embedding Integration with Relational Graph Convolutional Network), which directly incorporates various types of collaborative relations, such as user–user and item–item interactions, into the embedding function for user preference modeling. Specifically, our model employs advanced graph convolutional network (GCN) techniques to integrate user–item, user–user, and item–item relations for comprehensive representation learning. EIR-GCN initially selects the most influential second-order neighbors from the user–item bipartite graph to form user–user and item–item connections. With these enriched connections, a message-passing method is adopted to learn node representations by aggregating messages from directly linked nodes, including first-order item neighbors and selected second-order user neighbors. Extensive experiments on several public datasets demonstrate that EIR-GCN outperforms strong baselines, including recent GCN-based models and those exploiting high-order information. Our results show that EIR-GCN achieves state-of-the-art performance and effectively addresses the sparsity issue, highlighting its robustness and efficacy in recommendation tasks.

Keywords: collaborative filtering; recommendation; graph convolutional network; high-order neighborhood proximity

check for updates

Citation: Xiao, B.; Chen, D. Explicitly Exploiting Implicit User and Item Relations in Graph Convolutional Network (GCN) for Recommendation. *Electronics* **2024**, *13*, 2811. https://doi.org/10.3390/electronics13142811

Academic Editor: Fabio Grandi

Received: 21 May 2024
Revised: 7 July 2024
Accepted: 13 July 2024
Published: 17 July 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Online consumption (e.g., online shopping, reading news, and watching movies) has become the first choice for an increasingly large number of users. Recommendation, which matches users with the most appropriate items to suit their preference, has become a key technique to assist users in quickly finding their desired products among the overwhelming choices available online. Various online platforms have deployed recommender systems, such as Amazon (https://www.amazon.com, accessed on 15 July 2024) and Netflix (https://www.netflix.com/hk/, accessed on 15 July 2024), to enhance user satisfaction and increase profits.

Model-based collaborative filtering (MCF) methods are widely used due to their ability to achieve reasonable performance by leveraging user–item interaction data without additional information. A common paradigm of MCF methods involves learning user and item embeddings (i.e., d-dimensional vectors) by reconstructing historical interactions. These learned embeddings are expected to capture user preferences and item features. For an unseen item, its representation and the user's representation are used to predict the

user's preference using an interaction function (e.g., inner product). A typical example of MCF is matrix factorization (MF) [1]. In recent years, significant advancements have been made to enhance the performance of MCF methods. Neural collaborative filtering models, for instance, have been developed to capture the nonlinear interactions between users and items [2–4]. Additionally, concerns have been raised about the inner product, a widely adopted interaction function in MCF models, as it does not satisfy the triangle inequality and therefore fails to capture fine-grained user preferences accurately [5]. To address this issue, metric learning-based approaches have been proposed, embedding user and item representations into a metric space (e.g., Euclidean space) [5–7], resulting in substantially improved performance over baselines that use the inner product as the interaction function.

Despite these advances, an inherent limitation of MCF methods remains unresolved: their performance deteriorates dramatically when interactions are sparse, a problem known as the sparsity issue. This is because MCF methods rely solely on user–item interactions to learn representations. Sparse interactions make it challenging to learn high-quality representations, leading to significant performance drops. The fundamental assumption of collaborative filtering is that users with similar preferences will like similar items. Users who like the same set of items likely share common interests, and items liked by many common users likely have similar features. Therefore, based on shared interactions, we can identify user–user and item–item pairs with high similarity and form associations between them to enhance representation learning. Implicit item–item and user–user relations have been leveraged in graph-based methods to improve user preference modeling. For instance, several methods have exploited high-order relations between users and items by performing random walks on the user–item bipartite graph, constructed from user–item interactions where users and items are treated as two types of nodes. High-order relations, such as user–item [8,9], item–item, or user–user relations [10], are used as additional training data for building recommendation models. These studies have demonstrated improved performance, highlighting the effectiveness of exploiting high-order collaborative signals. However, the performance of these methods heavily depends on the quality of random walks, which require careful selection and tuning [11]. Additionally, sampled data from random walks may introduce noisy information, especially when sampled from long random walks.

In contrast to random walk-based methods, graph convolutional network (GCN)-based methods have demonstrated the ability to directly exploit high-order relations in the user–item bipartite graph for user and item embedding learning [12,13]. The core of GCN-based methods is to iteratively aggregate information from local neighbors to update node embeddings. By stacking multiple layers, information from high-order neighbors is also incorporated into the embedding learning process. This direct exploitation of high-order neighbor information distinguishes GCN-based recommendation models from methods that use high-order relations as external training data or regularization [13,14]. GCN-based recommendation methods have set new performance standards on benchmark datasets [13–15] due to their powerful embedding learning capabilities. However, they also face several limitations. Firstly, stacking many layers can lead to the over-smoothing problem, where the convolution operation in GCNs, a form of Laplacian smoothing, mixes features of local neighbors with the target node. This results in node representations becoming overly similar. Therefore, it is crucial to ensure that graph convolution operates within clusters of similar nodes to maintain distinctive and informative embeddings [16].

In this paper, we propose a novel graph-based recommendation model, EIR-GCN, which leverages additional collaborative information extracted from the user–item bipartite graph, specifically user–user and item–item relations, to enhance performance and alleviate data sparsity issues [10,17]. Unlike previous methods, our model constructs user–user and item–item connections using only second-order neighbors. These connections are directly incorporated into the model's embedding function. In the bipartite graph constructed from user–item interactions, for each node, the most similar second-order neighbor nodes are selected by a simple yet effective algorithm to form additional user–user and item–item

connections. This selection method is based on the assumption that nodes sharing more common neighbors are more likely to have similar preferences (for user–user pairs) or features (for item–item pairs) (see Section 3.2 for details). Using second-order neighbors instead of higher-order ones helps avoid introducing noisy information, which can impair performance. (=In experiments, we demonstrate that not all high-order neighbors are beneficial to performance (see Section 5.3)). With the newly constructed connections, we apply advanced graph convolution network (GCN) techniques to learn user and item embeddings. Specifically, the message-passing strategy [18,19] is employed to learn node representations by aggregating messages from directly linked nodes (i.e., first-order item neighbors and newly connected second-order user neighbors). This approach allows the model to directly leverage second-order neighbors for embedding learning. By applying GCN techniques to exploit second-order user and item neighbors, our model, EIR-GCN, effectively integrates additional user–user and item–item information, addressing the data sparsity problem for users with limited interactions.

The aim of this paper is to address the issue of sparsity in collaborative filtering-based recommender systems by proposing a novel model EIR-GCN. This model directly incorporates various types of collaborative relations, such as user–user and item–item interactions, into the embedding function for user preference modeling. By leveraging advanced graph convolutional network (GCN) techniques, EIR-GCN integrates user–item, user–user, and item–item relations for comprehensive representation learning. The primary goal is to enhance the robustness and efficacy of recommendation tasks, especially in scenarios where user and item interactions are sparse. Extensive experiments were conducted on three large-scale real-world datasets to validate the effectiveness of EIR-GCN. We carefully evaluated our model and compared it with various state-of-the-art baselines, including recently proposed models that also exploit high-order neighbors, such as LightGCN [13], HOP-Rec [8], and CSE [10]. The experimental results demonstrate the superiority of our model over these strong baselines and its capability in tackling the sparsity problem. In summary, the main contributions of this work are as follows.

- We highlight the advantages of leveraging second-order neighbors in the user–item bipartite graph to enhance recommendation performance.
- We propose an easy-to-implement model, EIR-GCN, which effectively exploits second-order neighbors in the embedding function to directly improve user and item representation learning. Additionally, we introduce a simple yet effective algorithm to select similar second-order neighbors to construct user–user and item–item connections.
- We conduct extensive empirical studies on three large-scale real-world datasets. The experimental results validate our assumptions and demonstrate the superior performance of EIR-GCN over several state-of-the-art methods. We release our code for reproducibility (https://github.com/Bwen-Xiao/EIR-GCN, accessed on 15 July 2024).

The rest of this paper is organized as follows: Section 2 provides an overview of the related work. Section 3 delves into the details of our EIR-GCN model. Following this, Section 4 presents the experimental setup and Section 5 discusses the results obtained from these experiments. The paper concludes with Section 6, summarizing the key findings and contributions.

2. Related Work

A comprehensive review of recommender systems is beyond the scope of this work. In this section, we primarily discuss recent advancements in model-based collaborative filtering (CF) models, especially focusing on graph-based models and graph convolutional network (GCN) techniques for recommendation, which are closely related to our work.

Matrix factorization (MF) [20] is a classical model-based CF method that has garnered significant research attention. MF maps users and items into a latent space, representing each user–item with a feature vector. These feature vectors are learned by reconstructing the interaction matrix based on the inner product of the feature vectors for each user–item pair. MF achieved notable success in the Netflix Prize Contest [21]. However, it has some limita-

tions: (1) the use of the inner product as the interaction function hinders performance [2,5], and (2) relying solely on user–item interactions leads to the data sparsity problem, where performance degrades when user–item interactions are sparse. To address these issues, many approaches have been proposed. Deep learning techniques have been widely applied to enhance the interaction function by introducing nonlinearity into the model, capturing the nonlinear interactions between users and items [2,3]. Metric learning approaches have been developed to use Euclidean distance to model interactions, addressing the issue that the inner product does not satisfy the triangle inequality and thus fails to capture fine-grained user preferences [5–7]. However, these approaches still struggle with the data sparsity problem.

A common and effective approach to mitigate the data sparsity problem is to leverage side information, which provides additional insights into user preferences and item features. Widely studied side information includes attribute labels [22,23] and review information [24–26]. Additionally, co-occurrence item–item (e.g., co-view) or user–user (e.g., friends) relations have been utilized to enhance preference modeling [17,27]. Another research direction to tackle the sparsity problem involves exploiting high-order proximity [8,11] and other types of relations, such as user–user and item–item interactions, in the user–item bipartite graph [9,10].

Given that our method falls into the category of leveraging high-order relations and is GCN based, we now turn to discussing graph-based and GCN-based models in more detail.

Graph-based CF models. Based on user–item interactions, a bipartite graph can be constructed by treating users and items as two types of nodes, linking them according to their interactions. Graph-based models can explicitly exploit high-order proximity between users and items. Early approaches inferred indirect preferences by performing random walks in the graph to provide recommendations [28–30]. Recently proposed approaches exploit the user–item bipartite graph to enrich user–item interactions [8,9] and explore other types of collaborative relations, such as user–user and item–item similarities [9,10]. For example, HOP-Rec [8] uses the random sampling of positive user–item interactions to enrich the training data through random walks. WalkRanker [9] and CSE [10] perform random walks to explore high-order proximity in user–user and item–item relations. As these methods rely on random walks to sample new interactions for model training, their performance heavily depends on the quality of interactions generated by random walks. Consequently, these methods require careful selection and tuning. Additionally, the high-order relations are used as additional information to regularize the original training in these methods.

Our model differs from these methods in two key aspects. Firstly, we leverage neighbors only from the second-order, enabling the use of a simple method to select positive user–user and item–item relations as additional information. Secondly, the additional data directly contribute to the representation learning function in our model, rather than serving merely as regularization information.

GCN-based recommendation. GCNs can naturally integrate node information and topological structure for representation learning [31,32]. Due to their powerful capability in representation learning, GCN techniques have attracted increasing attention in recommendation systems [18,33,34] and have been applied to various recommendation tasks [19,35]. For example, Ying et al. [15] designed an efficient GCN-based recommendation method, PinSage, which combines efficient random walks and graph convolutions to learn node embeddings and has been successfully applied for web-scale image recommendation. Berg et al. [18] proposed a novel graph auto-encoder framework, GCMC, which learns representations using a single graph convolutional layer in the encoder by exploiting direct connections between users and items, with a bilinear decoder used to reconstruct these connections. More recently, Wang et al. [11] proposed to explicitly exploit collaborative signals from high-order neighbors into the embedding function, designing a new model called NGCF, which achieves state-of-the-art recommendation performance. However, all these models rely on the original structure of the user–item bipartite graph and do

not explicitly exploit user–user and item–item relations in the graph. In contrast, our method creates new connections between similar user–user and item–item pairs, which are directly used for node representation learning. LightGCN [13] distinguishes itself as a simplified and effective GCN model by eliminating the transformation matrix and non-linear activation function, focusing solely on neighbor aggregation. UltraGCN [36] takes this concept further by avoiding explicit message-passing, thereby simulating the effect of infinite message-passing layers. GTN [37] introduces a novel approach to graph trend collaborative filtering, proposing a new graph trend filtering network that adaptively captures the reliability of user–item interactions. The recent integration of advanced techniques like disentangled learning and self-supervised learning into GCN models marks another significant advancement in recommender systems. This fusion has led to the development of more powerful GCN-based recommendation models as demonstrated by [16,38–42]. These innovative approaches have greatly enhanced the capabilities of recommendation systems, highlighting the continuous evolution and adaptability of GCN applications in this field. However, despite these advancements, a common limitation remains: these models primarily depend on single-behavior user–item interaction data. This reliance often results in data sparsity, posing a challenge to the effectiveness and accuracy of the recommendation process.

3. Our Proposed Model

3.1. Problem Setting and Model Overview

3.1.1. Problem Setting

Before describing our method, we would like to first introduce the problem setting. Given an interaction matrix $\boldsymbol{R}$ of dimensions $N_u \times N_v$, where N_u and N_v are the sizes of the user set $\mathcal{U}$ and the item set $\mathcal{V}$, respectively, a nonzero entry $r_{u_i v_j}$ in the matrix indicates that a user $u_i \in \mathcal{U}$ has interacted with an item $v_j \in \mathcal{V}$, and a zero entry means that there is no interactions between them. Notice that the interactions can be implicit (e.g., clicks) or explicit (e.g., ratings). Our goal is to learn a recommendation model that can recommend a user with suitable items that this user has not interacted with in the past. In the model-based recommendation method, this is achieved by first learning a vector representation for each user $u_i \in \mathcal{U}, i \in \{1, \ldots, N_u\}$ and each item $v_j \in \mathcal{V}, j \in \{1, \ldots, N_v\}$. With the user and item representations ($\boldsymbol{p}_{u_i}$ and $\boldsymbol{q}_{v_j}$, respectively), the preference score $\hat{r}_{ij}$ of u_i towards v_j can be predicted based on $\boldsymbol{p}_{u_i}$ and $\boldsymbol{q}_{v_j}$ with an interaction function (e.g., inner product). The top n items with the highest scores are then recommended to the user. In this work, we refer to the graph convolutional network techniques to learn user and item representations.

The interaction matrix can be represented by an undirected graph $\mathcal{G} = (\mathcal{W}, \mathcal{E})$, where $\mathcal{W}$ denotes the set of nodes and $\mathcal{E}$ is the set of edges. Specifically, $\mathcal{W}$ consists of user nodes $u_i \in \mathcal{U}$ with $i \in \{1, \ldots, N_u\}$ and item nodes $v_j \in \mathcal{V}$ with $j \in \{1, \ldots, N_v\}$, such that $\mathcal{U} \cup \mathcal{V} = \mathcal{W}$. When an interaction exists between two nodes u_i and v_j, there will be an edge $e_{u_i v_j} \in \mathcal{E}$ to link the two nodes in the graph. As interactions only exist between different types of nodes, there are only connections between user nodes and item nodes. Therefore, the constructed graph based on the interaction matrix is a bipartite graph as shown in Figure 1a.

3.1.2. Model Overview

The proposed EIR-GCN model consists of two stages: *second-order neighbor selection* and *representation learning*. In the first stage, the most similar users (or items) for a target user (or item) are selected from the second-order neighbors in the user–item bipartite graph. A significant advantage of exploiting only second-order neighbors, instead of higher-order relations, is that it simplifies the design of the neighbor selection algorithm, which does not require careful tuning to avoid introducing noisy information. After selection, new connections are constructed between users or items based on the selected second-order neighbors and the corresponding target node, forming a new user–item graph. Figure 1 illustrates this process with a toy example. In the figure, (a) shows the original user–item

bipartite graph; (b) demonstrates the selection of second-order neighbors; and (c) displays the new graph with user–user and item–item connections based on the selection results. In the second stage, a GCN method is employed for node representation learning in the new graph.

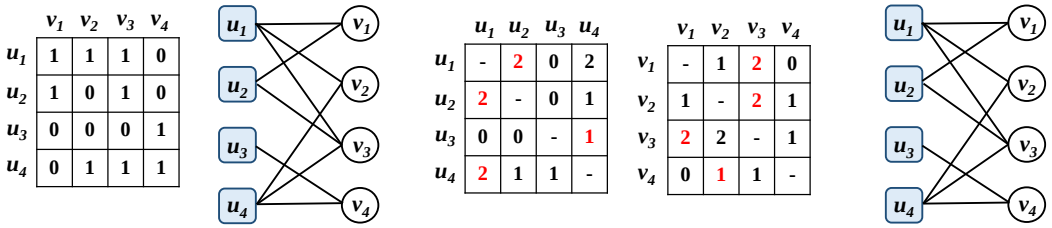


Figure 1. Illustration of exploiting second-order neighbors in a user–item bipartite graph to construct user–user and item–item connections. The first graph shows the original user–item bipartite graph based on the original user–item interaction matrix. A value in the constructed user–user and item–item matrix indicates the number of paths between two nodes. In this example, the second-order neighbors with the most number of paths are selected, as indicated by the values in red. For user u_1 and item v_3, since there are two neighbors with the same number of paths, one is selected randomly. The second graph shows the new graph, which adds user–user and item–item connections based on the selection results.

3.2. Second-Order Neighbor Selection

We aim to select the second-order neighbors that can contribute to the representation learning of users and items. The intuition is that preference information from similar users can be leveraged to learn a target user's preferences, and features from similar items are beneficial for an item's representation learning. Therefore, the goal of our selection method is to choose nodes with *high similarities* to the target node from the second-order neighbors.

Using random walks in the user–item bipartite graph may sample dissimilar users or items, introducing noisy information into the representation learning. To avoid this problem, we design a simple and reliable method for node selection based on two assumptions. First, *the shorter the path from one node to another, the higher the probability that the two nodes are similar*. The path here indicates the shortest path between two nodes in the graph. This is intuitive: a user has more similar preferences with users sharing first-order neighbors than those only sharing higher-order neighbors. For example, user A and user B like the movie *Iron Man* because they both like Marvel comics, while user B and user C like the movie *Final Destination* because they both enjoy horror movies. Although user A and user C are high-order neighbors, they may not share any common interests. Thus, we only select second-order neighbors to construct user–user and item–item connections, ensuring the use of relevant information.

The second assumption is that *the more common first-order neighbors two nodes have, the more similar the two nodes are*. A neighbor node that shares only a few commonly interacted items with a target node offers limited useful information for learning the target node's representation. Therefore, we select the second-order neighbors that share the most common neighbors with a node. Additionally, interactions with very popular items may not reflect a user's true preferences, as users might engage with popular items for social reasons. Hence, we select the second-order neighbors with the highest number of paths (with two hops) to the target node, effectively identifying those that share the most first-order neighbors with the target node. Although simple, our selection method effectively identifies similar user–user and item–item pairs for constructing connections for representation learning.

For each node, we select only the top k most similar second-order neighbors, where k is usually small, such as 30 in our experiments. This ensures scalability and flexibility in learning user and item representations from large-scale datasets. A large k risks introducing noise, particularly for nodes with sparse first-order neighbors. For a node with sparse interactions, the number of shared common neighbors with second-order neighbors becomes

sparse. A large k may select second-order neighbors with only one or two common neighbors, which may not be similar to the target node, resulting in performance degradation as verified in our experiments (see Section 5.3.2).For nodes with sparse first-order neighbors (i.e., users or items with sparse interactions), adding k (user–user or item–item) interactions can significantly enrich their associated information and enhance representation learning. For nodes with rich interactions, relatively good representations can already be learned based on the original user–item interactions, reducing the benefits of additional connections. This viewpoint is demonstrated in our experiments (see Section 5.2).

It is worth mentioning that we also tested an alternative neighbor selection strategy. This strategy selects a second-order neighbor w' for a target node w based on the ratio of the number of common neighbors to the total number of different neighbors, namely, $|\mathcal{N}_w \cap \mathcal{N}_{w'}|/|\mathcal{N}_w \cup \mathcal{N}_{w'}|$, where $\mathcal{N}_w$ is the first-order neighbor set of node w and $|\mathcal{N}_w|$ is its size. The top second-order neighbors with the largest ratios are selected. This strategy can avoid the undesirable situation where nodes with many neighbors (e.g., users who interact with many items) are more likely to be the top k second-order neighbors based on the first method. For instance, user A interacts with 1000 items, including 20 items that user B interacts with, while user C interacts with 20 items, including 18 items that user B interacts with. Based on the first selection strategy, user A ranks higher than user C in user B's second-order neighbors. However, user C should share more similar preferences with user B than user A because their interaction histories are almost identical. The second selection strategy avoids such cases. However, in practice, it does not show improved performance over the first strategy, likely because instances similar to the given example are sparse in real datasets. Therefore, we retain the first method due to its simplicity.

3.3. Representation Learning via GCN

The items consumed by a user directly provide information about their preferences, and the preferences of other users who have consumed many of the same items also encode valuable information about that user's preferences. Similarly, the consumption patterns of a group of users can be used to profile an item, and two items that have been consumed by many common users should share similar characteristics. Based on the user–item graph with the newly added user–user and item–item connections, we aim to leverage information from both user–item interactions and these new connections to learn user and item representations via GCN.

In the context of recommendation systems, graph convolutional networks (GCNs) are used to model the relationships between users and items represented as a bipartite graph. This graph consists of two types of nodes, users and items, connected by edges that signify interactions (e.g., user u has interacted with item i). Briefly, GCNs operate through a process known as **message-passing** and aggregation:

- **Message-Passing**: In this phase, each node (user or item) sends its information (features or embeddings) to its neighbors. For example, a user node will pass its embedding to all the item nodes it has interacted with, and vice versa.
- **Aggregation**: Each node then collects (aggregates) the information received from its neighbors to update its own embedding. This aggregation process combines the features from the neighboring nodes, allowing the model to capture the high-order connectivity and the collaborative filtering signals in the graph.

In the message-passing phase, valuable information is filtered from a node to its local neighbor nodes. During the message aggregation phase, each node aggregates all the information received from its neighbor nodes to update its own embedding. This approach allows for the collaborative leveraging of information from both the original user–item interactions and the newly constructed user–user or item–item connections for user and item representation learning. Through multiple layers of message-passing and aggregation, GCNs iteratively refine the embeddings of users and items, capturing complex interaction patterns and enhancing the recommendation performance by leveraging both direct and indirect relationships in the user–item bipartite graph.

Without loss of generality, we describe our algorithm using user representation learning as an example. The representations of items can be obtained in the same way.

3.3.1. Message-Passing

In our graph, a user node u is now connected to both *items* which are the *1-order neighbors* in the original bipartite graph, and other *users* which are the selected *2-order neighbors* from the original bipartite graph. Before describing the algorithm, we first define some notations. $\boldsymbol{p_u} \in \mathbb{R}^d$ and $\boldsymbol{q_v} \in \mathbb{R}^d$ are the embeddings of the user u and item v, respectively. d is the embedding size. $\mathcal{N}_{uv}$ denotes the 1-order item neighbors, and $\mathcal{N}_{uu}$ denotes the 1-order user neighbors in the new graph.

Message from items. The message from an item $v \in \mathcal{N}_{uv}$ to a target user u is defined as:

$$\boldsymbol{m}_{u \leftarrow v} = \gamma_v^u \boldsymbol{q}_v \tag{1}$$

where $\boldsymbol{m}_{u \leftarrow v}$ is the message embedding from the item v to the user u. γ_v^u is a parameter to control how much information is to be passed from user u to item v, which is computed by an attention mechanism that will be introduced later.

Message from users. As we construct user–user connections, the information from similar users can be also exploited to learn user preferences. Similar to the message from items, the message from a linked user $u' \in \mathcal{N}_{uu}$ to a target user u is defined as:

$$\boldsymbol{m}_{u \leftarrow u'} = \gamma_{u'}^u \boldsymbol{p}_{u'} \tag{2}$$

where $\boldsymbol{m}_{u \leftarrow u'}$ denotes the message embedding from user u' to user u. $\gamma_{u'}^u$ denotes the control of the amount of information passed from user u' to user u. Similarly, it is computed by an attention mechanism described in the following.

3.3.2. Attention Mechanism

We assume that different item nodes have different influence on user u. Inspired by this assumption, we design an attention mechanism to estimate the influence of item nodes on the user. The influence from item v to user u is formulated as follows:

$$s_v^u = g(\boldsymbol{p_u}, \boldsymbol{q_v}), \tag{3}$$

where $g(\cdot)$ is a similarity function to measure the similarity of vectors, which represents the relation of passing message from the item node to the user node. In particular, the cosine similarity function is applied in our work. We normalize the weight s_v^u to obtain the contribution of each item to user representation:

$$\gamma_u^v = \frac{\exp(s_v^u)}{\sum_{v' \in \mathcal{N}_v^u} \exp(s_{v'}^u)}. \tag{4}$$

where $\mathcal{N}_v^u$ denotes the set of items connected to user u. The normalized weight γ_u^v represents the contribution of item v to the user's representation, ensuring that items with higher similarity to the user have a larger influence.

Similarly, the contribution of each linked user to another user's representation $\gamma_u^{u'}$ can be obtained in the same way, allowing the model to consider both user–item and user–user interactions in the embedding learning process. By incorporating this attention mechanism, the model can selectively emphasize the most relevant interactions, leading to more accurate and personalized recommendations. This mechanism enhances the model's ability to capture nuanced preferences and relationships in the user–item bipartite graph.

3.3.3. Message Aggregation

In the second phase, the message passed from both the item neighbors and user neighbors are aggregated to update the user's embedding. Specifically, the aggregation function is defined as

$$\boldsymbol{p_u} = \boldsymbol{m}_{u \leftarrow u} + \lambda_\alpha \sum_{v \in \mathcal{N}_{uv}} \boldsymbol{m}_{u \leftarrow v} + \lambda_\beta \sum_{u \in \mathcal{N}_{uu'}} \boldsymbol{m}_{u \leftarrow u'} \tag{5}$$

where λ_α and λ_β are parameters trading off the amount of information passed from the item v and the linked user u'. Similar to previous work [11,18]. $m_{u \leftarrow u} = \boldsymbol{p}_u$ denotes the original embedding of the user.

3.3.4. Discussion

In our model, we leverage the message-passing method to exploit second-order neighbors selected from the original user–item bipartite graph. The original user–item interactions and the constructed user–user and item–item relations are treated similarly but contribute differently to the representation learning due to different learnable weight matrices.

The specific weights in Equation (5) are crucial to ensure that both first-order (direct user–item) and second-order (user–user and item–item) connections contribute appropriately to the embedding process. The learnable weights λ_α and λ_β are crucial, as they determine the relative importance of the information coming from different types of connections. They are trained to optimize the model's performance on the recommendation task. During training, the optimization algorithm adjusts these weights to find the best balance that maximizes the accuracy of user and item embeddings. By tuning λ_α and λ_β, the model can emphasize more on the direct user–item interactions if they provide more accurate signals or on the user–user/item–item relations if they offer additional valuable context, especially in scenarios with sparse interactions. Note that the weights are initialized and then iteratively updated during the training process. The model learns to assign higher weights to more informative connections, ensuring that both types of connections (first-order and second-order) contribute appropriately. The normalization in the attention mechanism (described in Equation (4)) further refines these contributions by ensuring that the influence of each neighbor is proportionate to its relevance as measured by the similarity function.

In summary, by specifying the weights λ_α and λ_β and allowing them to be learnable parameters, the model dynamically balances the contributions of different types of connections. This approach ensures that both direct and indirect relationships are appropriately leveraged to enhance the representation learning process, leading to more accurate and effective recommendations.

3.4. Prediction

With the learned embeddings of users (i.e., $\boldsymbol{p}_u$) and items $\boldsymbol{q}_v$, given a user u and a target item v, the preference of the user to the item is computed by inner product:

$$\hat{r}_{uv} = \boldsymbol{p}_u^T \boldsymbol{q}_v. \tag{6}$$

Notice that other interaction functions can be also applied, such as Euclidean distance. Because the main focus of this work is to study the effects of exploiting user–user and item–item relations for recommendation, we adopt the inner product as previous work [8,10,11] for fair comparisons in the empirical studies.

3.5. Optimization

3.5.1. Objective Function

In this work, we target the top n recommendation, which aims to recommend a set of n to ranked items which match the target user's preferences. Compared to the rating

prediction, this is a more practical task in real commercial systems [43]. Similar to other rank-oriented recommendation work [8,11], we adopt the pairwise-based learning method for optimization. To perform the pairwise learning, it needs to constructs a triplet of a user u, a positive item v^+, and a negative item v^-, with *an observed interaction between u and* v^+ and *an unobserved interaction between u and* v^-. This method assumes that a positive item (i.e., v^+) should rank higher than a negative item (i.e., v^-). The objective function is formulated as:

$$\arg\min \sum_{(\mathbf{u},\mathbf{v}^+,\mathbf{v}^-)\in\mathcal{O}} -\ln \phi(\hat{r}_{uv^+} - \hat{r}_{uv^-}) + \lambda\|\Theta\|_2^2 \quad (7)$$

where $\mathcal{O} = \{(u, v^+, v^-)|(u, v^+) \in \mathcal{R}^+, (u, v^-) \in \mathcal{R}^-\}$ denotes the training set; $\mathcal{R}^+$ indicates the observed interactions between user u and v^+ in the training dataset, and $\mathcal{R}^-$ is the sampled unobserved interaction set. λ and Θ represent the regularization weight and the parameters of the model, respectively. ϕ is the sigmoid function. The L_2 regularization is used to prevent overfitting.

3.5.2. Model Training

We implement our algorithm with the matrix form propagation rule (see [11] for more details), by which we can simultaneously update the representations of all users and items in a rather efficient way. With this implementation, we can discard the node sampling procedure. It is a commonly used approach to make the graph convolution network feasible for a large-scale graph [11,44]. Notice that besides the addition of new edges into the graph, the training of our model is of no difference to the training of the NGCF model with 1 propagation layer (i.e., **NGCF**$_1$) [11]. Thus, it is much simpler than NCGF with multiple layers. For each node, we at most add k connections, and k is small (e.g., 30 in our experiments). Therefore, our model can be trained efficiently and is applicable to large-scale datasets.

The mini-batch Adam [45] is adopted to optimize the prediction model and update the model parameters. Specifically, for a batch of randomly sampled triples $(u, v^+, v^-) \in (O)$, the representation of those users and items are first learned by the propagation rules, and then the model parameters are updated by using the gradients of the loss function.

Message and node dropout. Deep learning models often suffer from overfitting, so both message dropout and node dropout techniques are adopted in our implementation. Dropout is an effective strategy to prevent overfitting in neural network models. Specifically, message dropout and node dropout have been successfully applied in previous GCN models [11,18]. Node dropout involves randomly discarding a particular node, thereby blocking all its outgoing messages. As a result, this node cannot contribute to the new representation learning, making the embeddings more robust against the presence or absence of specific user or item influences. Message dropout, on the other hand, randomly drops individual outgoing messages, which can be considered edge dropout. This technique ensures the model becomes less dependent on specific edges, promoting more generalized learning. The drop ratios for message dropout (ρ_m) and node dropout (ρ_n) are empirically tuned in practice to achieve optimal performance.

4. Experimental Setup

4.1. Datasets

We used the public Amazon dataset (http://jmcauley.ucsd.edu/data/amazon, accessed on 5 March 2024) in our experiments. This dataset contains user interactions with products from Amazon and is organized into different product categories. For evaluation, we selected four datasets: *Toys and Games*, *Kindle Store*, *Home and Kitchen*, and *Movies and TV*. We followed the general practice in recommendation systems to filter out users and items with very few interactions. Specifically, we used the 10-core setting for all datasets, retaining only users and items with at least 10 interactions. The statistics of the three datasets are shown in Table 1. For each dataset, we randomly split it into training, validation, and testing sets with an 80:10:10 ratio for each user. The observed user–item interactions

were treated as positive instances. For methods using the pairwise learning strategy, we randomly sampled a negative instance (i.e., an item the user did not consume) to pair with each positive instance.

Table 1. Basic statistics of the experimental datasets. #ave.$_{user}$ and #ave.$_{item}$ represent average number of users per item and average number of items per user, respectively.

Dataset	#User	#Item	#ave.$_{user}$	#ave.$_{item}$	#Interactions	Sparsity
Toys and Games	19,412	11,924	14.06	8.63	167,596	99.93%
Kindle Store	14,356	15,885	23.13	25.60	367,477	99.84%
Home and Kitchen	66,519	28,237	19.54	8.29	551,681	99.97%
Movies and TV	33,326	21,901	43.79	28.78	958,985	99.87%

4.2. Experimental Settings

4.2.1. Evaluation Metrics

In this work, we focus on the top-n recommendation task, which aims to recommend a set of n top-ranked items that will be appealing to the target user. The following two widely used evaluation metrics are used in the evaluation:

- **Hit Ratio (HR)**: It indicates the percentage of users that have at least one correctly recommended item in their lists. It evaluates how likely it is that the recommendation system will provide at least one good recommendation to different users.
- **NDCG** [46]: This measure takes the positions of correctly recommended items into considerations. As users usually only focus on the top few results in a recommendation list, it is important to rank the correct ones at the top positions.

For each evaluation metric, the performance is evaluated based on the top 10 results. We report the average value across all users in the test set.

4.2.2. Baselines

To demonstrate the effectiveness, we compare our proposed EIR-GCN model with a variety of methods, including the classic matrix factorization-based model (BPR [43]), deep learning-based model (NeuMF [2]), recently proposed GCN-based models (GCMC [18], NGCF [11], and LightGCN [13]), and the ones exploiting high-order neighbors (HOP-Rec [8], CSE [10], and NGAT4Rec [47]).

- **BPR:** Bayesian Personalized Ranking (BPR) combines the matrix factorization method with the pairwise learning-to-rank loss function. And it has been proven to be a competitive baseline for top n recommendations [2,25].
- **NeuMF:** This method generalizes the matrix factorization to neural networks. It adopts multiple neural layers on top of the elementwise and the concatenation of user and item embeddings to capture their nonlinear interactions. It is a state-of-the-art neural CF model.
- **HOP-Rec:** This method exploits the high-order user–item interactions by random walks to enrich the original training data. In experiments, we used the codes released by the authors (https://github.com/cnclabs/smore, accessed on 3 April 2024).
- **CSE:** This recently proposed graph-based model also exploits the high-order proximity in the user–item bipartite graph. Different from HOP-Rec, this methods explore the user–user and item–item relations by random walks to improve the performance. We used the codes released by the authors (in the same link as HOP-Rec).
- **GCMC:** This method applies the GCN techniques on the user–item bipartite graph and employs one convolutional layer to exploit the direct connections between users and items.
- **NGCF:** It is a GCN-based recommendation model, which employs multi-layer GCN to leverage the collaborative signals in the form of high-order connectivities by performing information propagation in the user–item bipartite graph. We used the

implementation codes released by the authors (https://github.com/xiangwang1223/neural_graph_collaborative_filtering, accessed on 7 April 2024).

- **LightGCN:** This model simplifies the NGCF model by removing the transformation matrix and nonlinear activation function, focusing exclusively on neighbor aggregation. This simplification makes the model more efficient while retaining its effectiveness. To validate the effectiveness of our approach, we compared EIR-GCN to LightGCN using only one layer (denoted as LightGCN$_1$) and LightGCN with the optimal number of layers (denoted as LightGCN$_m$). **LightGCN**$_1$ serves as a baseline to directly compare the impact of using a single layer, as EIR-GCN also uses a single-layer approach. **LightGCN**$_m$ provides a benchmark for the best achievable performance with LightGCN, ensuring a comprehensive comparison. We used the implementation codes released by the authors (https://github.com/gusye1234/LightGCN-PyTorch, accessed on 7 April 2024).
- **NGAT4Rec:** The Neighbor-Aware Graph Attention Network for Recommendation (NGAT4Rec) improves recommendation by leveraging multi-hop neighbor information in the user–item interaction graph. It employs a novel neighbor-aware graph attention mechanism, which assigns different attention weights to neighbors based on pairwise attention, allowing for more granular relational information. NGAT4Rec aggregates embeddings using these attention weights and avoids feature transformation and nonlinear activation, enhancing collaborative filtering performance. This method consistently outperforms state-of-the-art models in recommendation tasks.

For fair comparisons, all the methods are optimized by the same pairwise learning strategy. We put great efforts into tuning these methods on the validation dataset and report their best performance.

4.2.3. Implementation Details

We implemented our model in Tensorflow. The embedding size was fixed to 64 for all models. For Hop-Rec and CSE, we searched the steps of random walks in $\{1, 2, 3, 4, 5\}$. The batch size of all models was fixed at 1024. We applied a grid search for hyperparameters: the learning rate was tuned amongst $\{0.0001, 0.0005, 0.001, 0.005\}$, the coefficient of L2 normalization was searched in $\{10^{-5}, \ldots, 10^{+2}\}$, and the dropout ratio in $\{0.0, 0.1, \ldots, 0.9\}$. The number of selected nodes (i.e., k) in EIR-GCN was searched in $\{10, 20, 30, 40, 50\}$. We also employed the node dropout technique for GC-MC, NGCF and EIR-GCN, where the ratio is tuned in $\{0.0, 0.1, \ldots, 0.9\}$. We used the Xavier initializer to initialize the model parameters. We adopted an early stop strategy which stops running if NDCG@20 does not improve for 50 successive epochs.

5. Experimental Results

5.1. Performance Comparisons

Table 2 reports the performance comparison results across four datasets. All reported results for EIR-GCN are based on $k = 30$, meaning we selected the top 30 second-order neighbors for model training. Notice that this setting is not optimal for all the datasets in our experiments. From the results, we have the following observations.

GCMC achieves better results than NeuMF across all datasets, demonstrating the advantages of GCN-based approaches in leveraging node information and graph structure. However, it is surpassed by models that more effectively utilize high-order connectivities, such as NGCF and LightGCN. CSE makes use of implicit associations of user–user and item–item similarities via high-order neighborhood proximity by performing random walks on the user–item bipartite graph. Therefore, CSE consistently outperforms NeuMF and GCMC across all cases, showing the utility of additional user–user and item–item relations in sparse data scenarios. HOP-Rec, which also uses high-order neighborhood information through random walks, generally performs well across all datasets but does not achieve the best performance. Its sampling approach enriches the training data effectively, but the reliance on random walks introduces variability.

Table 2. Performance of our EIR-NCG model and that of the competitors over three datasets. Notice that the values are reported by percentage with '%' omitted. The best and second best results are highlighted in bold.

Datasets	**Toys and Games**			**Kindle Store**			**Home and Kitchen**			**Movies and TV**		
Metrics	**Recall**	**HR**	**NDCG**	**Recall**	**HR**	**NDCG**	**Recall**	**HR**	**NDCG**	**Recall**	**HR**	**NDCG**
NeuMF	2.13	2.90	1.52	3.38	10.23	5.00	0.83	1.31	0.66	1.42	5.84	2.62
CSE	8.22	10.11	5.73	5.22	15.21	6.95	1.20	1.74	0.96	3.39	11.08	5.70
HOP-Rec	8.58	10.67	5.99	5.43	15.40	7.01	1.23	1.81	0.99	3.34	11.44	5.71
GCMC	3.38	4.57	2.32	5.41	15.37	6.97	0.88	1.41	0.68	2.52	9.44	3.89
NGCF	9.28	11.58	6.48	6.61	17.97	9.02	1.81	2.63	1.32	4.03	13.08	6.17
LightGCN_1	9.02	11.30	6.26	6.97	19.11	9.61	1.77	2.57	1.36	4.22	13.39	6.32
LightGCN_m	**9.81**	**12.16**	**6.79**	**7.75**	**20.63**	**10.77**	**2.00**	**2.92**	**1.55**	**4.44**	**14.01**	**6.79**
NGAT4Rec	8.17	10.22	5.65	5.05	14.56	7.56	1.95	2.89	1.56	3.30	11.22	5.51
EIR-GCN	**10.38 ***	**12.83 ***	**7.26 ***	**7.84 ***	**21.35 ***	**10.84 ***	**2.36 ***	**3.40 ***	**1.84 ***	**4.57 ***	**14.27 ***	**6.91 ***
Improv.	5.81%	5.51%	6.92%	1.16%	3.49%	0.65%	18.00%	16.44%	18.71%	2.93%	1.86%	1.77%

The symbol * denotes that the improvement is significant with p-value < 0.05 based on a two-tailed paired t-test.

NGAT4Rec leverages multi-hop neighbor information using a neighbor-aware graph attention mechanism, assigning different attention coefficients to various neighbors based on pairwise attention. This method captures more granular relational information and consistently performs well, although it does not outperform NGCF and LightGCN. NGCF consistently performs well across all datasets, leveraging multi-layer GCNs to capture high-order collaborative signals directly within the embedding process. This direct utilization of high-order information gives NGCF an edge over models like GCMC, which only use first-order neighbors, and those like HOP-Rec and LightGCN, both in its one-layer (LightGCN_1) and optimal-layer (LightGCN_m) configurations, performs exceptionally well. LightGCN_m achieves the best performance among the baselines on several datasets due to its simplified architecture that removes nonessential components, focusing purely on neighbor aggregation. This validates the effectiveness of a streamlined GCN model in recommendation tasks.

EIR-GCN outperforms all the baselines consistently across all datasets. The representation learning process of EIR-GCN is similar to NGCF with one embedding propagation layer but does not rely on multi-layer propagation. The superior performance of EIR-GCN demonstrates the effectiveness of directly leveraging user–user and item–item relations within the embedding function. This direct utilization yields substantial improvements over CSE and HOP-Rec, which also exploit high-order similarities but through more complex and less direct methods.

Overall, the results highlight several key takeaways: the performance of NeuMF on denser datasets underscores the importance of modeling nonlinear interactions between users and items. The improvements by CSE and HOP-Rec on sparser datasets demonstrate the effectiveness of leveraging high-order neighbor information to mitigate data sparsity. GCMC shows the advantages of GCN-based models in effectively utilizing node information and graph structure but is surpassed by models that directly integrate high-order connectivities. NGCF validates the power of capturing high-order collaborative signals directly within the embedding process, consistently outperforming models that use high-order information as auxiliary data. LightGCN further enhances this approach by streamlining the architecture and focusing purely on neighbor aggregation, achieving the best performance among the baselines on several datasets. Finally, the superior performance of EIR-GCN across all datasets demonstrates the significant advantage of directly incorporating additional collaborative relations, such as user–user and item–item connections, into the embedding process. The ability of EIR-GCN to leverage both user–user and item–item connections directly in the embedding function results in consistent and

substantial improvements over the baselines, highlighting its robustness and effectiveness in various recommendation scenarios.

5.2. Performance with Respect to Sparsity Level

To demonstrate the capability of EIR-GCN in handling users with limited interactions, we conducted experiments to study the performance of our method and other competitors across user groups with different sparsity levels. For each dataset, users were clustered into four groups based on their number of interactions in the training data. Taking "Toys and Games" as an example, users were divided into four groups based on the number of interactions: less than 10, 20, 30, and more than 30. Figure 2 shows the performance in terms of HR@10, Recall@10, and NDCG@10, for different user groups across four datasets. These figures also indicate the number of users in each group, showing that most users have fewer than 30 interactions, highlighting the common sparsity problem in real datasets. From the results, we derive the following observations:

EIR-GCN and LightGCN consistently achieve better performance than all other baselines across all user groups, followed by NGCF and other models leveraging high-order neighbors. This demonstrates that high-order information is highly beneficial for recommendation, particularly when directly exploited in the representation learning function, as performed by EIR-GCN and LightGCN. Additionally, CSE outperforms NeuMF when interactions are extremely sparse, such as fewer than 10 in the Kindle Store, showcasing the effectiveness of leveraging other types of collaborative signals to address the sparsity problem.

Analyzing the performance improvements across different user groups in the three datasets, we find that our model achieves more significant improvements in the first two groups compared to the best baselines. This verifies that user–user and item–item relations are particularly beneficial for the representation learning of inactive users. When interactions become relatively abundant, the benefits of additional information diminish, as the available user–item interactions suffice to learn good representations. Consequently, the performance of EIR-GCN becomes comparable to LightGCN in the fourth group across all datasets.

Overall, these results highlight the robustness and effectiveness of EIR-GCN in improving recommendation performance, especially for users with limited interactions.

5.3. Performance with Respect to Neighbor Selection

In the embedding learning function, EIR-GCN leverages user–user and item–item relations, which are constructed based on the top k nodes selected from second-order neighbors. In this section, we verify the effectiveness of our node selection algorithm and the influence of the number of selected nodes (i.e., k).

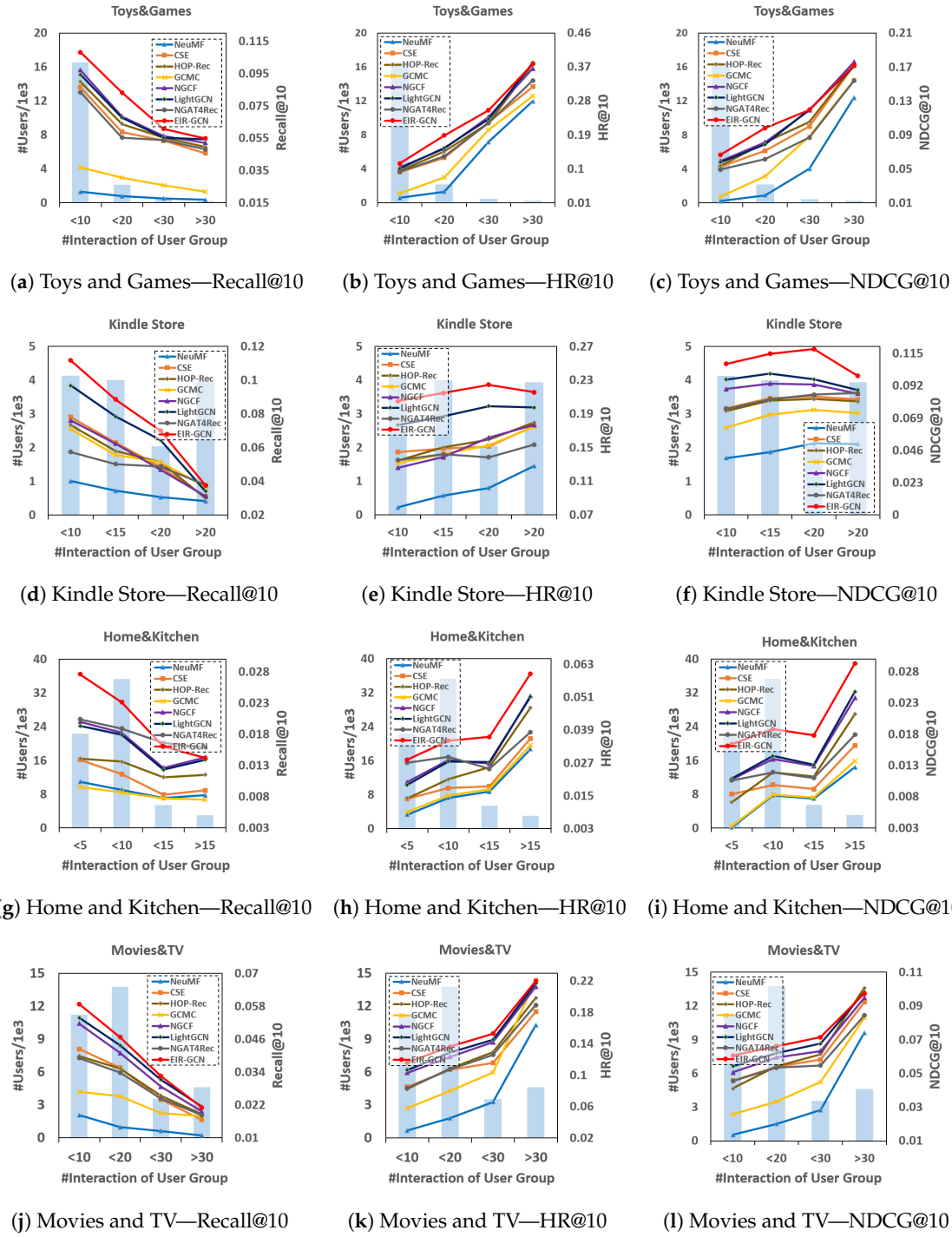
5.3.1. Effects of Neighbor Selection

To demonstrate the effectiveness of our node selection method, we compare EIR-GCN with two variants:

- **EIR-GCN**$_0$: This method does not leverage any second-order neighbors in the learning process.
- **EIR-GCN**$_r$: This method selects the second-order neighbors by random sampling.

Table 3 reports the performance of these three methods. The results for EIR-GCN and EIR-GCN$_r$ are obtained based on 30 neighbor nodes (i.e., $k = 30$). The substantial improvement in EIR-GCN over EIR-GCN$_0$ demonstrates the significant advantage of exploiting second-order neighbors in representation learning. We observe that EIR-GCN$_r$ outperforms EIR-GCN$_0$ on *Kindle Store* in terms of Recall and HR but yields inferior performance across all other cases. This indicates that random sampling may introduce noisy information to the system, resulting in performance degradation. Therefore, a reliable selection method is necessary to effectively extract positive signals from high-order

neighbors. The consistently superior performance of EIR-GCN over the two variants validates the effectiveness of our selection algorithm.



(**a**) Toys and Games—Recall@10 (**b**) Toys and Games—HR@10 (**c**) Toys and Games—NDCG@10

(**d**) Kindle Store—Recall@10 (**e**) Kindle Store—HR@10 (**f**) Kindle Store—NDCG@10

(**g**) Home and Kitchen—Recall@10 (**h**) Home and Kitchen—HR@10 (**i**) Home and Kitchen—NDCG@10

(**j**) Movies and TV—Recall@10 (**k**) Movies and TV—HR@10 (**l**) Movies and TV—NDCG@10

Figure 2. Performance comparison over the sparsity distribution of user groups on different datasets. The background histograms indicate the number of users involved in each group, and the lines show the performance.

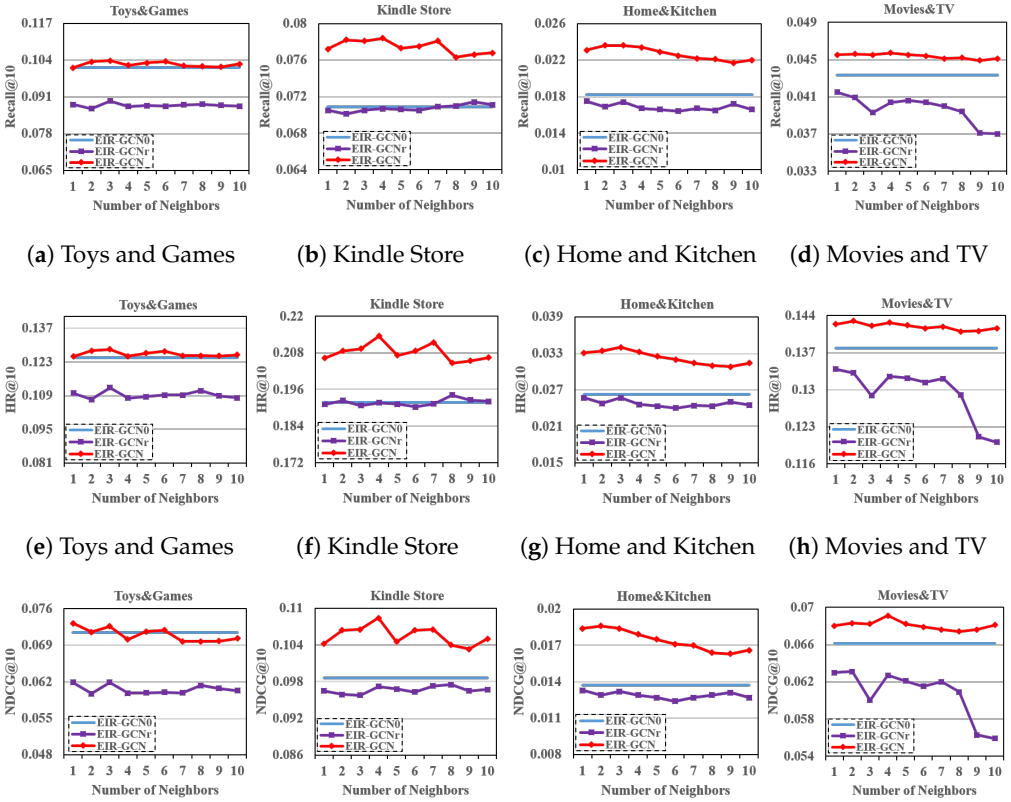
In summary, the results confirm that incorporating second-order neighbors selected by a reliable algorithm significantly enhances the representation learning process, leading to better recommendation performance.

Table 3. Effects of neighbor selection on performance. The best results are highlighted in bold.

Datasets	Toys and Games			Kindle Store			Home and Kitchen			Movies and TV		
Metrics	**Recall**	**HR**	**NDCG**	**Recall**	**HR**	**NDCG**	**Recall**	**HR**	**NDCG**	**Recall**	**HR**	**NDCG**
EIR-GCN$_0$	10.13	12.47	7.14	7.09	19.18	9.86	1.82	2.63	1.37	4.33	13.79	6.61
EIR-GCN$_r$	8.84	11.11	6.13	7.14	19.26	9.65	1.75	2.57	1.33	4.15	13.39	6.30
EIR-GCN	**10.38**	**12.83**	**7.26**	**7.84**	**21.35**	**10.84**	**2.36**	**3.40**	**1.84**	**4.57**	**14.27**	**6.91**

5.3.2. Effects of Neighbor Numbers

In this subsection, we analyze the impact of the number of selected neighbors on the performance of EIR-GCN. Figure 3 shows the performance of EIR-GCN by selecting different numbers of second-order neighbor nodes across different datasets. We compare three variants—EIR-GCN, EIR-GCN$_0$, and EIR-GCN$_r$—with the number of neighbors k ranging from 1 to 10. We mainly analyze the results based on Recall@10, as similar trends are observed for other two metrics. From Figure 3a–d, we observe that EIR-GCN consistently outperforms EIR-GCN$_0$ and EIR-GCN$_r$ across all datasets. The performance of EIR-GCN$_0$ remains flat, indicating that not leveraging second-order neighbors significantly limits the model's ability to capture complex user–item interactions. EIR-GCN$_r$, which randomly selects second-order neighbors, shows varying performance, sometimes even worse than EIR-GCN$_0$, especially in the *Movies and TV* dataset, demonstrating that random sampling may introduce noise. The performance of EIR-GCN generally increases with the number of neighbors up to a point, after which it stabilizes or slightly decreases, suggesting that an optimal number of neighbors exists that balances informativeness and noise.



(**a**) Toys and Games (**b**) Kindle Store (**c**) Home and Kitchen (**d**) Movies and TV

(**e**) Toys and Games (**f**) Kindle Store (**g**) Home and Kitchen (**h**) Movies and TV

(**i**) Toys and Games (**j**) Kindle Store (**k**) Home and Kitchen (**l**) Movies and TV

Figure 3. Performance vs. # Selected Nodes.

The analysis of neighbor numbers reveals that the performance of EIR-GCN benefits significantly from incorporating second-order neighbors. However, there is an optimal

number of neighbors, typically around 7-8, beyond which the performance gains stabilize or slightly decrease due to potential noise introduction. This indicates the importance of a reliable selection method to extract valuable information from high-order neighbors. The consistent better performance of EIR-GCN over its variants across various metrics and datasets validates the effectiveness of our neighbor selection algorithm and its integration into the embedding learning process.

6. Conclusions

In this work, we presented a novel EIR-GCN model for top-n recommendation. It explicitly incorporates collaborative signals from user–user and item–item relations into the embedding function of model-based collaborative filtering (CF). Specifically, a simple yet effective method is designed to select similar second-order neighbors in the user–item bipartite graph to form user–user and item–item connections. The message-passing method is then applied to gather information from similar users and items for representation learning. Consequently, the proposed EIR-GCN can naturally exploit additional collaborative information from second-order neighbors for representation learning, akin to the original user–item interactions.

Extensive experiments have been conducted on four real-world datasets. The results demonstrate that our method outperforms a variety of strong baselines, highlighting the potential of directly using user–user and item–item relations in the embedding learning process. This improvement is particularly significant for users with limited interactions, where the additional collaborative information from second-order neighbors greatly enhances the representation learning and recommendation performance. Overall, the EIR-GCN model demonstrates the importance and effectiveness of incorporating additional collaborative signals directly into the embedding function, providing a robust and scalable solution for enhancing recommendation systems in sparse data scenarios.

In the future, we would like to investigate more efficient methods for neighbor selection and message-passing to enhance the model's applicability to large-scale systems. Moreover, exploring the integration of other types of relations, such as social connections and contextual information, could provide a richer context for recommendation and further improve performance. Finally, developing adaptive methods for hyperparameter tuning, such as automatically determining the optimal k, can reduce the need for empirical study and make the model more user friendly. These directions offer promising avenues for future research and development, aiming to further improve the effectiveness and efficiency of recommendation systems.

Author Contributions: Conceptualization, supervision, writing—review and editing, D.C.; methodology, validation, formal analysis, writing—original draft preparation, B.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 42–49. [CrossRef]
2. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International World Wide Web Conference, Perth, Australia, 3–7 April 2017; pp. 173–182.
3. Xue, H.J.; Dai, X.Y.; Zhang, J.; Huang, S.; Chen, J. Deep matrix factorization models for recommender systems. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 3203–3209.
4. Cheng, Z.; Ding, Y.; He, X.; Zhu, L.; Song, X.; Kankanhalli, M.S. A^3NCF: An Adaptive Aspect Attention Model for Rating Prediction. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 3748–3754.
5. Hsieh, C.K.; Yang, L.; Cui, Y.; Lin, T.Y.; Belongie, S.; Estrin, D. Collaborative metric learning. In Proceedings of the 26th International World Wide Web Conference, Perth, Australia, 3–7 April 2017; pp. 193–201.

6. Tay, Y.; Anh Tuan, L.; Hui, S.C. Latent relational metric learning via memory-based attention for collaborative ranking. In Proceedings of the Web Conference, Lyon, France, 23–27 April 2018; pp. 729–739.
7. Liu, F.; Cheng, Z.; Sun, C.; Wang, Y.; Nie, L.; Kankanhalli, M.S. User Diverse Preference Modeling by Multimodal Attentive Metric Learning. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 1526–1534.
8. Yang, J.; Chen, C.; Wang, C.; Tsai, M. HOP-rec: High-order proximity for implicit recommendation. In Proceedings of the 12th ACM Conference on Recommender Systems, Vancouver, BC, Canada, 2–7 October 2018; pp. 140–144.
9. Yu, L.; Zhang, C.; Pei, S.; Sun, G.; Zhang, X. WalkRanker: A Unified Pairwise Ranking Model With Multiple Relations for Item Recommendation. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 2596–2603.
10. Chen, C.; Wang, C.; Tsai, M.; Yang, Y. Collaborative Similarity Embedding for Recommender Systems. In Proceedings of the Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 2637–2643.
11. Wang, X.; He, X.; Wang, M.; Feng, F.; Chua, T. Neural Graph Collaborative Filtering. In Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 165–174.
12. Liu, F.; Chen, H.; Cheng, Z.; Nie, L.; Kankanhalli, M. Semantic-Guided Feature Distillation for Multimodal Recommendation. In Proceedings of the 31st ACM International Conference on Multimedia, Ottawa, ON, Canada, 29 October–3 November 2023; pp. 6567–6575.
13. He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; Wang, M. 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, Virtual Event, China, 25–30 July 2020; pp. 639–648.
14. Chen, L.; Wu, L.; Hong, R.; Zhang, K.; Wang, M. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 27–34.
15. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 18–23 August 2018; pp. 974–983.
16. Liu, F.; Cheng, Z.; Zhu, L.; Gao, Z.; Nie, L. Interest-aware message-passing gcn for recommendation. In Proceedings of the Web Conference, Ljubljana, Slovenia, 12–16 April 2021; pp. 1296–1305.
17. Liang, D.; Altosaar, J.; Charlin, L.; Blei, D.M. Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Co-occurrence. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 59–66.
18. van den Berg, R.; Kipf, T.N.; Welling, M. Graph Convolutional Matrix Completion. In Proceedings of the KDD 2018 Deep Learning Day, London, UK, 19–23 August 2018.
19. Wei, Y.; Cheng, Z.; Yu, X.; Zhao, Z.; Zhu, L.; Nie, L. Personalized Hashtag Recommendation for Micro-videos. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 1446–1454.
20. Koren, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Las Vegas, Nevada, USA, 24–27 August 2008; pp. 426–434.
21. Bell, R.M.; Koren, Y. Lessons from the Netflix prize challenge. *SIGKDD Explor.* **2007**, *9*, 75–79. [CrossRef]
22. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 7–10.
23. Chen, J.; Zhuang, F.; Hong, X.; Ao, X.; Xie, X.; He, Q. Attention-driven Factor Model for Explainable Personalized Recommendation. In Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, Washington DC, USA, 14–18 July 2018; pp. 909–912.
24. Cheng, Z.; Ding, Y.; Zhu, L.; Mohan, K. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In Proceedings of the Web Conference Lyon, France, 23–27 April 2018; pp. 639–648.
25. Zhang, Y.; Ai, Q.; Chen, X.; Croft, W.B. Joint representation learning for top-n recommendation with heterogeneous information sources. In Proceedings of the 10th ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017; pp. 1449–1458.
26. Guan, X.; Cheng, Z.; He, X.; Zhang, Y.; Zhu, Z.; Peng, Q.; Chua, T. Attentive Aspect Modeling for Review-Aware Recommendation. *ACM Trans. Inf. Syst.* **2019**, *37*, 28:1–28:27. [CrossRef]
27. Yang, X.; Guo, Y.; Liu, Y.; Steck, H. A survey of collaborative filtering based social recommender systems. *Comput. Commun.* **2014**, *41*, 1–10. [CrossRef]
28. Gori, M.; Pucci, A. ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines. In Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 6–12 January 2007; pp. 2766–2771.
29. Christoffel, F.; Paudel, B.; Newell, C.; Bernstein, A. Blockbusters and Wallflowers: Accurate, Diverse, and Scalable Recommendations with Random Walks. In Proceedings of the 9th ACM Conference on Recommender Systems, Vienna, Austria, 16–20 September 2015; pp. 163–170.

30. Fouss, F.; Pirotte, A.; Renders, J.; Saerens, M. Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation. *IEEE Trans. Knowl. Data Eng.* **2007**, *19*, 355–369. [CrossRef]
31. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
32. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How Powerful are Graph Neural Networks? In Proceedings of the 7th International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
33. Guo, L.; Liu, H.; Zhu, L.; Guan, W.; Cheng, Z. DA-DAN: A Dual Adversarial Domain Adaption Network for Unsupervised Non-overlapping Cross-domain Recommendation. *ACM Trans. Inf. Syst.* **2023**, *42*, 48. [CrossRef]
34. Wang, H.; Zhang, F.; Zhang, M.; Leskovec, J.; Zhao, M.; Li, W.; Wang, Z. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 968–977.
35. Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, Y.E.; Tang, J.; Yin, D. Graph Neural Networks for Social Recommendation. In Proceedings of the Web Conference 2019, San Francisco, CA, USA, 13–17 May 2019; pp. 417–426.
36. Mao, K.; Zhu, J.; Xiao, X.; Lu, B.; Wang, Z.; He, X. UltraGCN: Ultra simplification of graph convolutional networks for recommendation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Queensland, Australia, 1–5 November 2021; pp. 1253–1262.
37. Fan, W.; Liu, X.; Jin, W.; Zhao, X.; Tang, J.; Li, Q. Graph trend filtering networks for recommendation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, 11–15 July 2022; pp. 112–121.
38. Wang, X.; Jin, H.; Zhang, A.; He, X.; Xu, T.; Chua, T.S. Disentangled graph collaborative filtering. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, China, 25–30 July 2020; pp. 1001–1010.
39. Wang, L.; Jin, D. A Time-Sensitive Graph Neural Network for Session-Based New Item Recommendation. *Electronics* **2024**, *13*, 223. [CrossRef]
40. Li, M.; Li, J.; Yang, L.; Ding, Q. Self-Supervised Hypergraph Learning for Knowledge-Aware Social Recommendation. *Electronics* **2024**, *13*, 1306. [CrossRef]
41. Cai, X.; Huang, C.; Xia, L.; Ren, X. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. *arXiv* **2023**, arXiv:2302.08191.
42. Cui, Y.; Zhou, P.; Yu, H.; Sun, P.; Cao, H.; Yang, P. ASKAT: Aspect Sentiment Knowledge Graph Attention Network for Recommendation. *Electronics* **2024**, *13*, 216. [CrossRef]
43. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; pp. 452–461.
44. Qiu, J.; Tang, J.; Ma, H.; Dong, Y.; Wang, K.; Tang, J. DeepInf: Social Influence Prediction with Deep Learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 2110–2119.
45. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
46. Järvelin, K.; Kekäläinen, J. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* **2002**, *20*, 422–446. [CrossRef]
47. Song, J.; Chang, C.; Sun, F.; Song, X.; Jiang, P. Ngat4rec: Neighbor-aware graph attention network for recommendation. *arXiv* **2023**, arXiv:2010.12256.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.