electronics

MDPI

Article

Preference-Aware Light Graph Convolution Network for Social Recommendation

Haoyu Xu [1], Guodong Wu [1], Enting Zhai [1], Xiu Jin [1] and Lijing Tu [2,*]

[1] College of Information and Computer Science, Anhui Agricultural University, Hefei 230001, China; xhy817@stu.ahau.edu.cn (H.X.)
[2] Anhui Provincial Key Laboratory of Smart Agricultural Technology and Equipment, Hefei 230036, China
* Correspondence: tlj@ahau.edu.cn

Abstract: Social recommendation systems leverage the abundant social information of users existing in the current Internet to mitigate the problem of data sparsity, ultimately enhancing recommendation performance. However, most existing recommendation systems that introduce social information ignore the negative messages passed by high-order neighbor nodes and aggregate messages without filtering, which results in a decline in the performance of the recommendation system. Considering this problem, we propose a novel social recommendation model based on graph neural networks (GNNs) called the preference-aware light graph convolutional network (PLGCN), which contains a subgraph construction module using unsupervised learning to classify users according to their embeddings and then assign users with similar preferences to a subgraph to filter useless or even negative messages from users with different preferences to attain even better recommendation performance. We also designed a feature aggregation module to better combine user embeddings with social and interaction information. In addition, we employ a lightweight GNN framework to aggregate messages from neighbors, removing nonlinear activation and feature transformation operations to alleviate the overfitting problem. Finally, we carried out comprehensive experiments using two publicly available datasets, and the results indicate that PLGCN outperforms the current state-of-the-art (SOTA) method, especially in dealing with the problem of cold start. The proposed model has the potential for practical applications in online recommendation systems, such as e-commerce, social media, and content recommendation.

Keywords: graph convolution network; recommendation system; social recommendation

Citation: Xu, H.; Wu, G.; Zhai, E.; Jin, X.; Tu, L. Preference-Aware Light Graph Convolution Network for Social Recommendation. *Electronics* **2023**, *12*, 2397. https://doi.org/10.3390/electronics12112397

Academic Editor: Grzegorz Dudek

Received: 18 April 2023
Revised: 15 May 2023
Accepted: 19 May 2023
Published: 25 May 2023

1. Introduction

With the emergence and prosperity of online service platforms, the dissemination and exchange of information have been greatly promoted, and the amount of information in the network has increased exponentially. However, when confronted with such an enormous amount of information, users find it hard to obtain the information that is relevant and helpful to them; this phenomenon is referred to as "information overload". To address this issue, a recommendation system was developed that analyzes the historical behavior data of users and explores their potential interests to provide them with personalized services. At present, recommendation systems are widely used in industry.

Collaborative filtering (CF) has been a widely used technique in the last few decades. In simple terms, collaborative filtering recommends information of interest to users according to the preferences of a group of people who share similar interests and experiences, thus filtering out a large amount of irrelevant information. However, CF is severely limited by the problem of sparse data, and the effectiveness of the model is significantly reduced when there are insufficient data on user–item interactions. As online social platforms such as Facebook, WeChat, and Twitter have grown in popularity, an increasing number of people are posting product reviews on these sites. References [1–3] and personal experience also

show that people are affected by their friends' views and actions and gravitate toward those who share their interests. In summary, the application of social relationships in recommender systems has also attracted increasing attention [4,5]. Based on this understanding, recommendation systems can introduce social information to reduce data sparsity and improve recommendation accuracy, and these recommendation systems are called social recommendation systems [6–8].

Early GNNs mainly solved problems strictly related to graph theory [9–11], such as molecular structure classification, in which GNNs showed a superb ability to handle non-Euclidean data. Since data in recommender systems can naturally represent graph data (e.g., interaction data between users and items represented as bipartite graphs), much recent work has applied graph neural networks to recommender systems. Within social recommendation systems, data are typically presented in two forms: the user–item interaction graph, which contains information pertaining to the interactions between users and items, and the social graph, which reflects the social relations of users. There are generally two strategies for recommender systems to use social information [12]. One is to learn user representations from the two graphs separately [8,13,14] and then combine them into a vector, which is more flexible and can use different treatments for different graphs; the other is to merge the two graphs into a unified heterogeneous graph [7,15] and apply GNNs to propagate the information, which has the advantage that the information in both graphs is unified in one representation, which can capture some more complex interactions.

Although GNNs have shown good performance in the field of social recommendation, most of the existing models simply combine social information as auxiliary information with interactive information without fully utilizing the social graph's information. In the information propagation, they only consider the information propagation of high-order neighbor nodes, but no particular attention is given to the fact that there is a lot of useless or even negative information in this information. Inspired by IMP-GCN [16], we introduce an unsupervised subgraph construction module in the social recommendation system, which divides the interaction graph into multiple subgraphs based on user preferences, and users who share similar interests are placed into the same subgraph. We then perform graph convolution operations in the subgraphs using a lightweight GNN to filter out negative information brought by users with different preferences. We also design a feature aggregation module to better integrate user representations in the two graphs.

In conclusion, the primary findings of this study are as follows:

- A novel social recommendation model PLGCN is proposed, which splits the user–item interaction graph into multiple subgraphs based on the user's preferences and passes information in the subgraphs, filtering out negative information brought by users with different preferences.
- A new feature aggregation module was designed that can aggregate the user representations in the two graphs more effectively and has regularization to prevent overfitting.
- We performed comprehensive experiments using two publicly available datasets to evaluate the recommended performance of PLGCN. Based on the outcomes of these experiments, it is evident that PLGCN outperforms the baseline models.

The rest of this article is summarized as follows: We begin with a brief overview of typical relevant work in Section 2. The social recommender system problem and its definition are introduced in Section 3. The design details of the PLGCN model are described in Section 4. Section 5 presents a comprehensive experiment conducted to assess the performance of PLGCN. Finally, we conclude our work and identify potential research directions.

2. Related Work

2.1. Social Recommendation

As online social platforms (e.g., WeChat, Twitter, Facebook) and the richness of users' social information grow rapidly, an increasing number of recommendation systems are introducing users' social information. Leveraging social influence [1] and social homogeneity,

as outlined in [2], facilitates a better comprehension of user preferences, and data sparsity is effectively mitigated.

We generally categorize the prior social recommendation systems relevant to our study into three groups based on how they utilize social information. Social networks are used as a kind of regularization in the first category of methods [4,17–19]. SocialMF [19] integrates trust propagation into the matrix factorization technique, making the user's preferences as close as possible to his/her social neighbors. CSR [18] designed a generic regularization term to model the diverse similarities among users and their various friends. One kind of ensemble method involves splitting all items into different groups and ranking them manually [20,21]. SBPR [20] suggests that users have a tendency to provide higher ratings to products that are favored by their friends, and for each user, the collection of items is sorted into three categories: negative items, social items, and positive items. The rankings are as follows: negative items < social items < positive items. The third method is to fuse the embeddings of the user and his/her neighbors [22–24]. TrustSVD [22] introduces social information based on SVD++ [25] and uses implicit feedback from social neighbors as auxiliary implicit feedback for users. RSTE [24] posits that the user's final choice is a trade-off between his/her own likes and the opinions of his/her trusted friends and the linear fusion of the user's embedding with the user's neighbor nodes in the social graph. Nevertheless, none of these models can adequately model the intricate social relationships between users and the interactions between users and items. Therefore, numerous recent studies have focused on employing deep learning for social recommendation systems, with GNN-based social recommendation systems attracting attention because both the social relationship between users and the data that reflect user–item interactions can be modeled naturally in graph form.

2.2. GNN-Based Social Recommendation

The ability of GNNs to model recursive social diffusion processes makes them increasingly popular in the domain of social recommendation. The primary tenet of GNNs is to iteratively collect surrounding node information to generate a more accurate representation of the target node and ultimately obtain a representation of each node.

The first social recommendation system model that uses graph neural networks is GraphRec [6], which combines the user's first-order neighbors' information from both graphs to learn the user's representation. DANSER [14] uses two graph attention networks to obtain user and item implicit representations from each of the two graphs and then combines them to predict users' ratings or preferences for items. HOSR [26] uses multi-step message propagation to encode higher-order social relationships in user embedding learning, refining user embeddings by capturing higher-order collaboration signals in the social graph. In contrast, our proposed PLGCN captures both higher-order collaboration signals in the interaction graph and social graph simultaneously and fuses them into the final user embedding using a feature aggregation module. DiffNet [8] fully leverages the high-order social neighborhood information of users and adds the vector of users' preferred items as an auxiliary vector to the vector representation of users, but this does not filter out the useless parts of the high-order information. On the basis of the DiffNet model, DiffNet++ [7] additionally exploits the high-order information in the interaction graph to optimize the representation of users and items and distinguishes the significance of the different neighbors using the attention mechanism, thus alleviating this problem. Unlike DiffNet++, we are inspired by IMP-GCN [16] to divide the user-interaction graph into multiple subgraphs based on users' preferences to avoid interaction between users with different preferences. Furthermore, we design a new feature aggregation model to obtain a more accurate user embedding. The research conducted on SocialLGN [27] is closely related to our own research. These researchers extended the LightGCN [28] framework and applied convolution operations on both the interaction graph and the social graph to improve the framework's effectiveness in handling social recommendation problems. The main differences between SocialLGN and our proposed PLGCN are as follows: (1) SocialLGN

aggregates all messages from neighboring nodes without considering their relevance. In contrast, our PLGCN includes a subgraph construction module, and we perform message passing in the subgraph to effectively filter out irrelevant information. (2) Our feature aggregation module uses MLP to fully explore the potential relationship between user interaction information and social information, whereas SocialLGN uses a linear transformation in the graph fusion module.

In summary, several approaches have been proposed to address the challenges of social recommendation. However, these approaches often have limitations, such as the difficulty of modeling complex user–item interactions and social relationships. Some approaches use attention mechanisms to distinguish the importance of neighboring nodes and filter out irrelevant information, but they often oversimplify the use of social information and do not fully exploit its value. We propose the PLGCN approach to overcome these challenges, which uses subgraph building blocks to filter out irrelevant information and fully leverages the relationship between social and interaction information through neural networks. Our approach provides a more nuanced modeling of the complex interaction between users and items and the social relationship, leading to improved recommendation accuracy and relevance.

3. Problem Definition

Essentially, the recommendation problem is to analyze the user's behavior data to predict the preferences of the user and then combine the data of the items in the system to calculate the items that could potentially appeal to the user and generate a recommendation list from them. However, users are only able to explicitly interact with a small fraction of items, which results in very sparse valid data. The social recommendation system introduces users' social information, which supplements the sparse and effective data and reduces data sparsity. It is clearly effective to use the homogeneity and the influence of social relationships to understand users' preferences.

In the paragraphs that follow, we define a GNN-based social recommendation system. The notations $\mathcal{I}$ and $\boldsymbol{\mathcal{U}}$ are used to represent the sets of items and users, and they have M and N elements, respectively (i.e., $|\mathcal{I}| = M$, $|\boldsymbol{\mathcal{U}}| = N$). In general, recommendation systems make use of two distinct types of data: social graphs and user–item interaction graphs. A description of these two graphs is given below.

The interaction behavior of users with items (e.g., views, rates, and clicks) is represented by the user–item interaction graph. The graph can be defined by triples $(u, y_{ui}, i | u \in \boldsymbol{\mathcal{U}}, i \in \mathcal{I})$ and is denoted by G_I, where y_{ui} represents the edge that connects user u to item i, and $y_{ui} > 0$ means that user u interacts with item i. On the other hand, there will not be any interaction between them if $y_{ui} = 0$. The notation $\boldsymbol{\mathcal{N}}_i^I$ means the collection of users who have explicit interaction with item i, and the notation $\boldsymbol{\mathcal{N}}_u^I$ indicates the collection of items with which user u has explicit interaction.

Users' social connections are represented in the social graph, which provides auxiliary information about the user (e.g., direct follower or undirected friendship). We represent the social graph as G_s, which has the triple form $\{(u, s_{uv}, v | u, v \in \boldsymbol{\mathcal{U}})\}$, where s_{uv} represents the relationship between users u and v, and $s_{uv} = 1$ means there is an observable social connection between users u and v, while $s_{uv} = 0$ indicates there is no connection between them. The symbol $\boldsymbol{\mathcal{N}}_u^S$ is used to denote the collection of users who have a social connection with user u.

Based on the aforementioned conditions, the social recommendation task is described as follows: given the social graph G_S and the user–item interaction graph G_I, the recommendation system should be able to predict the probability of interaction between user u and all items, sort them in descending order, and choose the top N items to generate a recommendation list for user u.

4. The Proposed Method

We present the general structure and technical details of PLGCN as well as the model's training process.

4.1. The Architecture of PLGCN

Figure 1depicts the general design of PLGCN. The model consists of 4 primary components: (i) an embedding layer that leverages the unique identifiers of users and items to initialize their representation vectors; (ii) a subgraph construction module, which constructs multiple subgraphs and groups users with common preferences into the same subgraph; (iii) propagation layers, which propagate the representations of users and items in both graphs; and (iv) a prediction layer that predicts the value of any edge between users and items using their final embeddings (i.e., the probability that user and item will interact).

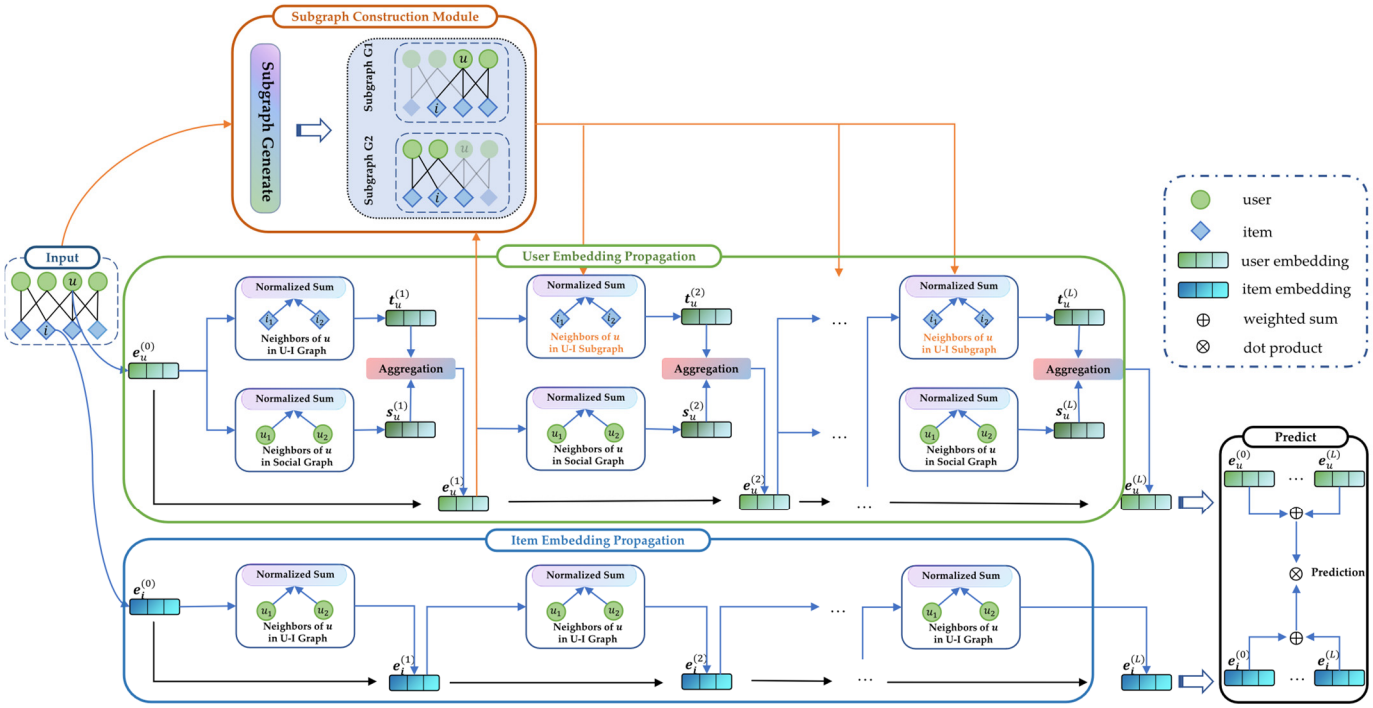

Figure 1. Architecture design of our PLGCN model with 2 subgraphs as illustration. First-order propagation operations are performed on the entire interaction graph and social graph, and higher-order propagation operations are performed on the subgraphs of the interaction graph and social graph.

The following is a description of the mechanism of operation of each component.

4.2. Embedding Layer

The embedding layer employs user and item identifiers to map them to a latent space with low dimensionality. As an example, the embedding layer can encode item i (or user u) as a low-dimensional dense vector of fixed length $\boldsymbol{e}_i^{(0)} \in \mathbb{R}^d$ (*or* $\boldsymbol{e}_u^{(0)} \in \mathbb{R}^d$), and the superscript "$(l)$" ($l \geq 0$) indicates the layer index for the output of the embeddings at the l-th propagation layer. When $l = 0$, it indicates the embedding layer's output. d is a hyperparameter determined in advance that indicates the dimension of an embedding.

The matrices $\boldsymbol{E}_U^{(0)} \in \mathbb{R}^{N \times d}$ and $\boldsymbol{E}_I^{(0)} \in \mathbb{R}^{M \times d}$ represent the output of all N users and all M items from the embedding layer, respectively. User u's embedding is the transpose of the matrix $\boldsymbol{E}_U^{(0)}$'s u-th row, and the same applies to item i.

4.3. Subgraph Construction Module

The subgraph construction module splits the given user–item interaction graph into N_c subgraphs, where the number of subgraphs N_c is a hyperparameter. We define the division of users into subgraphs as a classification task [29], in which each user is assigned to a subgraph. Specifically, each user's embedding aggregates graph structure information and the user's ID information:

$$\boldsymbol{F}_u = \sigma(\boldsymbol{b}_1 + \boldsymbol{W}_1\left(\boldsymbol{e}_u^{(0)} + \boldsymbol{e}_u^{(1)}\right)) \tag{1}$$

where $\boldsymbol{F}_u$ denotes the user embedding obtained by embedding aggregation, $\boldsymbol{e}_u^{(0)}$ represents the output generated by the embedding layer, and $\boldsymbol{e}_u^{(1)}$ is the feature vector that aggregates first-order neighbor information in the graph, which is generated as an output from the first propagation layer. σ denotes an activation function called LeakyReLU, capable of encoding both negative and positive signals. The learnable parameters $\boldsymbol{b}_1 \in \mathbb{R}^{1\times d}$ and $\boldsymbol{W}_1 \in \mathbb{R}^{d\times d}$ are the bias vector and the weight matrix, respectively. To split the user–item interaction graph into multiple subgraphs based on user preferences, we input the user embeddings into a 2-layer neural network to obtain a prediction vector:

$$\boldsymbol{U}_h = \sigma(\boldsymbol{b}_2 + \boldsymbol{W}_2\boldsymbol{F}_u) \tag{2}$$

$$\boldsymbol{U}_o = \sigma(\boldsymbol{b}_3 + \boldsymbol{W}_3\boldsymbol{U}_h) \tag{3}$$

where $\boldsymbol{U}_o$ is the output vector, and the position where the value is at its maximum is the subgraph to which the user belongs, so it is natural that the number of subgraphs and the output vector's dimension are the same. The learnable parameters $\boldsymbol{W}_2 \in \mathbb{R}^{d\times d}$ and $\boldsymbol{W}_3 \in \mathbb{R}^{1\times N_c}$ are the weight matrices, and the learnable parameters $\boldsymbol{b}_2 \in \mathbb{R}^{1\times d}$ and $\boldsymbol{b}_3 \in \mathbb{R}^{1\times N_s}$ are the bias vectors. For users with similar embeddings, the neural network will group them into the same subgraph. This is an unsupervised node classification method because we do not need the real labels of the users.

In summary, we feed the user ID information and first-order user embedding, which best reflect user preferences, into the subgraph construction module. Then, we utilize the powerful modeling ability of neural networks to handle nonlinear relationships and classify user preferences. It is worth noting that we refrain from using traditional clustering algorithms such as K-means [30] due to the high dimensionality of user feature vectors in the current recommendation system field. Traditional clustering algorithms are susceptible to the curse of dimensionality when dealing with high-dimensional data, which can lead to information loss if PCA-based [31] dimensionality reduction is used. Additionally, traditional clustering algorithms cannot effectively model complex nonlinear relationships.

The subgraph construction module groups users with similar preferences and their directly related items into the same subgraph, with each subgraph being independent. By passing messages only within each subgraph, our approach effectively filters out irrelevant or negative information.

4.4. Propagation Layers

The propagation layer aims to capture hidden information in both graphs through graph convolution operations, thereby learning the representations of users and items. The propagation layer is divided into two main parts: user embedding propagation and item embedding propagation. A detailed explanation of them is presented in the following subsections.

4.4.1. User Embedding Propagation

The goal of user embedding propagation is to learn the representation of users in both graphs. We use lightweight GNNs to capture the collaboration signals of the interaction graph and the social graph, propagate information on the two graphs separately, and finally

generate the final user embeddings through the feature aggregation module. The process of user u's l-th ($l \leq L$) iteration propagation can be abstracted as follows:

$$\boldsymbol{e}_u^{(l)} = Agg\left(\left\{\boldsymbol{e}_i^{(l-1)}, \forall i \in N_u^I\right\}, \left\{\boldsymbol{e}_v^{(l-1)}, \forall v \in N_u^S\right\}\right) \tag{4}$$

where $\boldsymbol{e}_i^{(l-1)}$ and $\boldsymbol{e}_v^{(l-1)}$ are the embeddings of item i and user v, respectively, after the l-th iteration propagation, and $Agg(*)$ is the aggregation function that aggregates the embeddings of item i with which u has interaction and the embeddings of user v with which u is socially connected. We designed a feature aggregation module to act as the user aggregation function $Agg(*)$ to better learn user embeddings.

Because direct interactions between users and items more accurately reflect user preferences, this is crucial and reliable information. To construct subgraphs based on user preferences, we perform first-order graph convolution operations on the social graph and entire interaction graph alone, while second-order or higher-order graph convolution operations are performed on the social graph and subgraphs of the interaction graph to filter out useless or even negative information from users with different preferences. To achieve this, two separate embeddings are created in the interaction graph and the social graph to represent user u after the l-th iteration propagation, with $\boldsymbol{t}_u^{(l)}$ and $\boldsymbol{s}_u^{(l)}$ being their respective representations and $\boldsymbol{e}_u^{(l)}$ being the user's final embedding after the l-th iteration propagation. Thus, for user u, the first-order propagation can be expressed as follows:

$$\boldsymbol{t}_u^{(1)} = \sum_{i \in \mathcal{N}_u^I} \frac{1}{c_{ui}} \boldsymbol{e}_i^{(0)} \tag{5}$$

$$\boldsymbol{s}_u^{(1)} = \sum_{v \in \mathcal{N}_u^S} \frac{1}{c_{uv}} \boldsymbol{e}_v^{(0)} \tag{6}$$

where c_{ui} is $\sqrt{|\mathcal{N}_u^I||\mathcal{N}_u^I|}$, which is the product of the square root of the degree of user u and item i in the interaction graph, and its inverse is the normalization term that prevents the user or item embedding scale from increasing due to graph convolution operations. c_{uv} is $\sqrt{\left|\mathcal{N}_u^S\right|\left|\mathcal{N}_v^S\right|}$, which is the product of the square root of the degree of user u and user v in the social graph and serves the same purpose as c_{ui}.

The process of updating the embedding in second-order or higher-order (i.e., $l \geq 2$) graph convolution is analogous to the first-order graph convolution process, with the difference that high-order graph convolution is performed in the social graph and the subgraph of the interaction graph to which the user belongs. The procedure is explained in full in the steps that follow:

$$\boldsymbol{t}_u^{(l)} = \sum_{i \in \boldsymbol{\mathcal{N}}_u^{I_c}} \frac{1}{c_{ui_c}} \boldsymbol{e}_i^{(l-1)} \tag{7}$$

$$\boldsymbol{s}_u^{(l)} = \sum_{v \in \boldsymbol{\mathcal{N}}_u^S} \frac{1}{c_{uv}} \boldsymbol{e}_v^{(l-1)} \tag{8}$$

where c_{ui_c} is $\sqrt{\left|\boldsymbol{\mathcal{N}}_u^{I_c}\right|\left|\boldsymbol{\mathcal{N}}_u^{I_c}\right|}$, which is the product of the square root of the degree of user u and item i in the subgraph of the interaction graph to which the user belongs. As Equations (5)–(8) show, we have adopted a lightweight form of propagation, discarding complex operations such as linear transformations, and this lightweight form of propagation is inspired by SGC [32] and LightGCN [28].

The feature aggregation module is then used to aggregate $\boldsymbol{t}_u^{(l)}$ and $\boldsymbol{s}_u^{(l)}$ to generate the updated embedding $\boldsymbol{e}_u^{(l)}$ for layer l. As shown in Equations (9) and (10), the feature

aggregation module can be seen as a function $Agg(*)$ with two embeddings as parameters, and the specific aggregation steps are as follows:

$$\boldsymbol{h}_u^{(l)} = MLP\left(\sigma\left(\boldsymbol{W}_4\boldsymbol{t}_u^{(l)}\right) \parallel \sigma\left(\boldsymbol{W}_5\boldsymbol{s}_u^{(l)}\right)\right) \tag{9}$$

$$\boldsymbol{e}_u^{(l)} = \frac{\boldsymbol{h}_u^{(l)}}{\left|\left|\boldsymbol{h}_u^{(l)}\right|\right|_2} \tag{10}$$

where $\boldsymbol{W}_4$ and $\boldsymbol{W}_5 \in \mathbb{R}^{d\times d}$ are trainable weight matrices and $\parallel$ is a vector splicing operation that splices two vectors of dimension d into a vector of length $2d$. σ is the *tanh* activation function, and $MLP(*)$ is a multilayer perceptron that can capture the complex relationship between two users' embeddings in each dimension. Equation (9) is a regularization operation that prevents embedding $\boldsymbol{e}_u^{(l)}$ from becoming particularly large as the number of layers l grows.

4.4.2. Item Embedding Propagation

For item embedding propagation, the propagation process is analogous to that of users, but this process exists only in the user–item interaction graph. We use lightweight GNNs to capture the collaboration signals and update the item embedding by recursively passing the representation of neighboring nodes. The specific process is shown as follows:

$$\boldsymbol{e}_i^{(l)} = \sum_{u\in\boldsymbol{\mathcal{N}}_i^I} \frac{1}{c_{iu}}\boldsymbol{e}_u^{(l-1)} \tag{11}$$

where c_{iu} is $\sqrt{|\mathcal{N}_u^I||\mathcal{N}_u^I|}$, which is the product of the square root of the degree of user u and item i in the interaction graph, and it also serves for normalization.

4.5. Prediction Layer

After K rounds of propagation, the embedding for user u and item i is obtained for each layer, i.e., $\left\{\boldsymbol{e}_u^{(0)}, \ldots, \boldsymbol{e}_u^{(L)}\right\}$ and $\left\{\boldsymbol{e}_i^{(0)}, \ldots, \boldsymbol{e}_i^{(L)}\right\}$, respectively. We weight and sum the user and item embeddings for each layer to obtain the final representation:

$$\boldsymbol{e}_u^* = \sum_{l=0}^{L} \alpha_l \boldsymbol{e}_u^{(l)} \tag{12}$$

$$\boldsymbol{e}_i^* = \sum_{l=0}^{L} \alpha_l \boldsymbol{e}_i^{(l)} \tag{13}$$

where α_l denotes the l-th layer's embedding weight factor and $\boldsymbol{e}_u^*$ and $\boldsymbol{e}_i^*$ are the final embeddings of user u and item i, respectively.

To obtain the preference of user u for item i, the inner product of their embeddings is computed:

$$\hat{y}_{ui} = \boldsymbol{e}_u^{*T}\boldsymbol{e}_i^* \tag{14}$$

where $\hat{y}_{ui}$ denotes our predicted preference of user u for item i.

4.6. Model Training

In general, the tasks of the recommender system are divided into two categories: CTR prediction and top-N recommendation. In this work, our recommendation task is top-N recommendation, where the aim is to select N items that suit the user's preferences best and recommend them to the user in the form of a list. In a real business system, this task is worth more than predicting ratings [33].

To achieve this, we minimize the Bayesian personalized ranking loss, which is based on the idea that it increases the gap between the scores of the negative and positive samples, with positive samples being user and item interactions that already exist in the dataset and negative samples being non-existent interactions that are not observed in the dataset. Therefore, we define a triple $\{u, i^+, i^-\}$, where u has interacted with i^+ but not with i^-. The objective function has the following form:

$$arg\ min \sum_{(u,i^+)\in\mathcal{N}_u^I \cup\ (u,i^-)\notin\mathcal{N}_u^I} -ln\sigma(\hat{y}_{ui^+} - \hat{y}_{ui^-}) + \lambda||\Theta||_2^2 \tag{15}$$

where λ and Θ denote the weight decay rate and the parameters of PLGCN, respectively.

4.7. Matrix-Form Propagation Rule of PLGCN

We propose a matrix-based representation of the PLGCN model for propagating information on graphs in this section. To achieve this goal, we use $\boldsymbol{R} \in \mathbb{R}^{N\times M}$ to represent the rating matrix. Each element $r_{ui}(u = 1 : N, i = 1 : M)$ inside the matrix is binary and shows if user u and item i have interacted; 1 implies that an interaction exists, and 0 indicates that there is no interaction. Then, $\boldsymbol{S} \in \mathbb{R}^{N\times N}$ indicates the adjacency matrix of social graph G_s.

We further define $\boldsymbol{\mathcal{L}}_R = \boldsymbol{D}_R^{-\frac{1}{2}}\boldsymbol{R}\boldsymbol{D}_R^{\frac{1}{2}}$, where $\boldsymbol{\mathcal{L}}_R$ is the Laplacian matrix for the interaction graph. $\boldsymbol{D}_R \in \mathbb{R}^{N\times N}$ is a diagonal matrix, where d_{ii} is the element that counts how many elements in $\boldsymbol{R}$'s i-th row are nonzero. In the same way, we also define the transpose matrix $\boldsymbol{\mathcal{L}}_R^T$ of $\boldsymbol{\mathcal{L}}_R$ and the Laplacian matrix for social graph $\boldsymbol{\mathcal{L}}_S$.

As illustrated in 0, the following expression describes the first layer propagation in PLGCN:

$$\boldsymbol{H}_U^{(1)} = MLP\Big(\sigma\Big(\boldsymbol{W}_1\boldsymbol{\mathcal{L}}_R\boldsymbol{E}_I^{(0)}\Big) \parallel \sigma\Big(\boldsymbol{W}_2\boldsymbol{\mathcal{L}}_S\boldsymbol{E}_U^{(0)}\Big)\Big) \tag{16}$$

$$\boldsymbol{E}_U^{(1)} = \frac{\boldsymbol{H}_U^{(1)}}{\boldsymbol{H}_U^{(1)}{}_2} \tag{17}$$

$$\boldsymbol{E}_I^{(1)} = \boldsymbol{\mathcal{L}}_R^T\boldsymbol{E}_U^{(0)} \tag{18}$$

The following formula shows the l-th layer's propagation in matrix form in PLGCN:

$$\boldsymbol{E}_{U_c}^{(l)} = \boldsymbol{\mathcal{L}}_c\boldsymbol{E}_{U_c}^{(l-1)} \tag{19}$$

where $\boldsymbol{\mathcal{L}}_c$ denotes the Laplacian matrix belonging to a subgraph of the interaction graph. The information of all subgraphs is then aggregated:

$$\boldsymbol{E}_U^{(l)} = \sum_{U_c\in G_c} \boldsymbol{E}_{U_c}^{(l)} \tag{20}$$

where $\boldsymbol{E}_U^{(l)}$ is the final embedding of the l-th layer, and G_c denotes the set of subgraphs of the user–item interaction graph. Then, we perform the same operation as the first layer:

$$\boldsymbol{H}_U^{(l)} = MLP\Big(\sigma\Big(\boldsymbol{W}_1\boldsymbol{\mathcal{L}}_R\boldsymbol{E}_I^{(l-1)}\Big) \parallel \sigma\Big(\boldsymbol{W}_2\boldsymbol{\mathcal{L}}_S\boldsymbol{E}_U^{(l-1)}\Big)\Big) \tag{21}$$

$$\boldsymbol{E}_U^{(l)} = \frac{\boldsymbol{H}_U^{(l)}}{\boldsymbol{H}_U^{(l)}{}_2} \tag{22}$$

$$\boldsymbol{E}_I^{(l)} = \boldsymbol{\mathcal{L}}_R^T\boldsymbol{E}_U^{(l-1)} \tag{23}$$

5. Empirical Analysis

To compare our PLGCN's performance with other recommendation methods, this section describes the evaluation metrics, the dataset, the parameter settings, and the experiments we carried out on various datasets. We ran all programs on a Win10 PC with an RTX 3070 Ti graphics card with 8 G of RAM and an i5 12,600 K processor. We used PyTorch to build the PLCGN.

5.1. Experimental Settings

5.1.1. Datasets

The proposed model was evaluated by conducting experiments on two datasets from the real world that vary in size and levels of sparsity. The datasets are described as follows: The LastFM dataset [34] (https://www.last.fm (accessed on 17 May 2023)), which includes 1892 users' social connections and interactions with music-related items, is provided by Last.fm, one of the hottest social music platforms of the moment for sharing and discovering music in the world. The Ciao dataset [35] (http://www.ciao.co.uk (accessed on 17 May 2023)) is an online shopping dataset that includes 7375 customer reviews for a variety of products as well as information about user friendships. Many social recommendation systems use these datasets to validate model performance [27,36,37]. We divided each dataset into three random parts, the training set, the test set, and the validation set, corresponding to a ratio of 8:1:1. The precise statistical data from the datasets we used are displayed in Table 1.

Table 1. Statistics for the two datasets. # represents the number of elements in the set.

Dataset	Ciao	LastFM
# Users	7375	1892
# Items	105,114	17,632
# Interactions	284,086	92,834
Density (Interactions)	0.037%	0.278%
# Social Connections	57,544	25,434
Density (Social Connections)	0.106%	0.711%

5.1.2. Benchmark Cases

We evaluated how well PLGCN performs by contrasting it with six other methods that are state-of-the-art:

- BPR [38]—A list of recommendations is created by sorting items using the traditional pairwise collaborative filtering approach according to the maximum posterior probability determined by a Bayesian analysis of the issue.
- SBPR [20]—An MF-based recommendation model to enhance the accuracy of personalized rankings with collaborative filtering algorithms using users' social relationships.
- DiffNet [8]—A social recommendation model that utilizes GNNs. It directly takes the user embeddings' vector sum in the two graphs to generate the final user embeddings.
- NGCF [39]—A recommendation model based on GCN is designed with a neural network approach to recursively propagate embeddings in the interaction graph.
- LightGCN [28]—A lightweight recommendation model based on GCN that eliminates two operations that would have caused recommendation performance degradation based on NGCF.
- SocialLGN [27]—User social information was introduced on the basis of LightGCN, and a graph fusion operation was created to combine user embeddings with interaction information and user embeddings with social information.

5.1.3. Metrics

To assess the recommended performance under the top-N task of our PLGCN and five other SOTA methods, we use three metrics that are commonly applied; two of them are precision and recall, and they have the following expressions:

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{24}$$

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{25}$$

where FP is the number of incorrectly predicted negative samples, TP denotes the number of properly predicted positive samples, and FN denotes the number of incorrectly predicted positive samples. The other is $NDCG$, i.e., normalized discounted cumulative gain, which is used to measure the quality of the ranking, and it is expressed as follows:

$$NDCG@N = \frac{r(1) + \sum_{i=2}^{l} \frac{r(i)}{log_2^i}}{\sum_{i=1}^{|REL|} \frac{r(i)}{log_2^{(i+1)}}} \tag{26}$$

where $|REL|$ is the sum of the relevance scores $r(i)$ of the top N items recommended. $r(i) = 1$ indicates that the user interacts with the recommended item; $r(i) = 0$ means that the user does not interact with the recommended item.

In summary, the indicators used in this experiment and their significance are listed below, and it is worth noting that these metrics are all dimensionless:

- Precision@k: the proportion of relevant items among the top k items recommended to the user. Precision@10 and Precision@20 indicate the precision at 10 and 20 recommendations, respectively.
- Recall@k: the proportion of relevant items among all the relevant items in the test set that are recommended to the user. Recall@10 and Recall@20 indicate the recall at 10 and 20 recommendations, respectively.
- NDCG@k: normalized discounted cumulative gain at k. NDCG is a measure of ranking quality that takes into account both the relevance of the recommended items and their position in the list. NDCG@10 and NDCG@20 indicate the NDCG score at 10 and 20 recommendations, respectively.

The greater the value for these three evaluation metrics, the better the performance. Given the sparsity of the interaction data, we repeatedly randomly selected an item that the user did not interact with as a negative sample; then, we combined items that the user did interact with the negative sample. To eliminate the instability of random selection, for each model and dataset, we repeated the experiment five times and averaged the results as the ultimate ranking results.

5.1.4. Parameter Settings

To ensure that the experiments were fair and equitable, the parameters of each method were adjusted based on our own experimental data or the corresponding references. We used the PyTorch framework to construct our PLGCN and Adam to infer model parameters. The model was optimized with a learning rate η of 1×10^{-3}. The dimension of embedding was fixed at 200, and the training batch size was fixed at 2048. After trials in the range of $\{1 \times 10^{-6}, 1 \times 10^{-5}, \ldots, 1 \times 10^{-2}\}$, we fixed weight decay (λ) at 1×10^{-4}. The weight (α_l) for each propagation layer was $\frac{1}{L+1}$, where L is the number of layers and N_c is the subgraph count. For this paper, L was set to 3, and the value of N_c was 2. Early stopping was adopted to terminate the training process. To enhance readability, we present the parameter settings in a tabular format, as shown in Table 2.

Table 2. Parameter settings.

Parameter	Value
Learning rate (η)	1×10^{-3}
Dimension of embedding (d)	200
Training batch size	2048
Weight decay (λ)	1×10^{-4}
Number of layers (L)	3
Number of subgraphs (N_c)	2

5.2. Model Performance Evaluation

In the recommendation field, the problem of cold start is a matter of great concern, for which we designed a special scenario to evaluate the cold-start performance of PLGCN and other baseline models by referring to the approach in the literature [27]. In both datasets, we compared the experimental results for all models, including the cold-start case. In the test set, users who interacted with fewer than 20 items were labeled as cold-start users. We employ this strategy to provide a test set for cold-start users alone, which includes only cold-start users and their social and interaction information. The results of PLGCN and the five baseline models on the original test set are shown in Table 3. The outcomes of the aforementioned models on the cold-start test set are displayed in Table 4, and the improvements in Tables 3 and 4 indicate the percentage increase in performance of our model in terms of precision, recall, and NDCG at 10 and 20 recommendations compared to the best baseline model. It is worth noting that all experimental results were not normalized.

Table 3. Recommendation performance of all models on both datasets. The underlined value is the second-best performance, and the bolded value is the best. Improvement is the comparison between the best performance and the second-best performance.

Dataset	Model	Precision@10	Precision@20	Recall@10	Recall@20	NDCG@10	NDCG@20
LastFM	BPR	0.0922	0.0720	0.0962	0.1499	0.1099	0.1321
	SBPR	0.1398	0.1010	0.1442	0.2070	0.1749	0.1978
	DiffNet	0.1727	0.1215	0.1779	0.2488	0.2219	0.2474
	NGCF	0.1766	0.1269	0.1796	0.2576	0.2287	0.2563
	LightGCN	0.1961	0.1358	0.2003	0.2769	0.2536	0.2788
	SocialLGN	<u>0.1972</u>	<u>0.1368</u>	<u>0.2026</u>	<u>0.2794</u>	<u>0.2566</u>	<u>0.2883</u>
	PLGCN	**0.2043**	**0.1412**	**0.2091**	**0.2881**	**0.2646**	**0.2903**
	Improvement	3.60%	3.22%	3.21%	3.11%	3.12%	0.69%
Ciao	BPR	0.0145	0.0111	0.0220	0.0339	0.0229	0.0260
	SBPR	0.0179	0.0141	0.0259	0.0412	0.0266	0.0307
	DiffNet	0.0238	0.0182	0.0341	0.0527	0.0359	0.0403
	NGCF	0.0178	0.0179	0.0343	0.0531	0.0359	0.0407
	LightGCN	0.0271	0.0202	0.0410	0.0591	0.0437	0.0478
	SocialLGN	<u>0.0276</u>	<u>0.0205</u>	<u>0.043</u>	<u>0.0618</u>	<u>0.0441</u>	<u>0.0486</u>
	PLGCN	**0.0308**	**0.0230**	**0.0446**	**0.0662**	**0.0476**	**0.0526**
	Improvement	13.59%	12.20%	3.72%	7.12%	7.94%	8.23%

The outcomes demonstrate that models based on MF do not perform as well in all cases and exhibit a performance much inferior to that of GNN-based models because MF-based models are more susceptible to data sparsity and cannot capture complex interactions. LightGCN performs better in the vast majority of cases than BPR, SBPR, DiffNet, and NGCF. As pointed out in [28], LightGCN removes two fundamental operations in GCN that can negatively affect recommendation performance, namely linear transformation and nonlinear activation. SocialLGN performs better than LightGCN because it introduces social information on top of LightGCN and considers the effect of higher-order graph structure on user embedding.

Table 4. Recommendation performance of all models on two cold-start datasets. The underlined value is the second-best performance, and the bolded value is the best. Improvement is the comparison between the best performance and the second-best performance.

Dataset	Model	Precision@10	Precision@20	Recall@10	Recall@20	NDCG@10	NDCG@20
LastFM-cold	BPR	0.0282	0.0209	0.1151	0.1615	0.0828	0.0989
	SBPR	0.0292	0.0333	0.1123	0.2467	0.0709	0.1159
	DiffNet	0.0417	0.0271	0.1713	0.2407	0.1107	0.1309
	NGCF	0.0333	0.0292	0.1169	0.2141	0.1074	0.1411
	LightGCN	0.0417	0.0313	0.1727	0.2416	0.1374	0.1560
	SocialLGN	0.0458	0.0333	0.1974	0.2663	0.1419	0.1643
	PLGCN	**0.0667**	**0.0396**	**0.2624**	**0.3000**	**0.1716**	**0.1821**
	Improvement	45.63%	18.92%	32.93%	12.65%	20.93%	10.83%
Ciao-cold	BPR	0.0061	0.0047	0.0208	0.0328	0.0138	0.0179
	SBPR	0.0070	0.0060	0.0234	0.0384	0.0165	0.0219
	DiffNet	0.0104	0.0081	0.0339	0.0539	0.0248	0.0316
	NGCF	0.0104	0.0085	0.0341	0.0557	0.0245	0.0319
	LightGCN	0.0131	0.0096	0.0429	0.0616	0.0319	0.0384
	SocialLGN	0.0134	0.0097	0.0441	0.0630	0.0328	0.0394
	PLGCN	**0.0144**	**0.0106**	**0.0447**	**0.0668**	**0.0336**	**0.0412**
	Improvement	7.46%	9.28%	1.36%	6.03%	2.44%	4.57%

The results unequivocally show that PLGCN consistently achieves the best performance. For instance, in contrast to SocialLGN, PLGCN improves the Recall@10 on the original LastFM dataset by 3.22% and the Precision@10 on the original Ciao dataset by 13.59%. Since SocialLGN propagates messages on the social graph and the whole user–item interaction graph without constructing subgraphs, by comparing the performance of PLGCN with SocialLGN in the experiments, it can be seen that propagating information in subgraphs can significantly raise the effectiveness of recommendations. In particular, on the LastFM dataset containing information about cold-start users only, PLGCN improves 45.63% in the Precision@10 metric and 32.93% and 20.93% in Recall@10 and NDCG@10, respectively. By looking at the data in 0, we can see the superior ability of PLGCN in alleviating the cold-start problem. Additionally, we find that in the cold-start scenario, the denser the interaction and social graphs of the dataset are, the more significant the performance improvement is, while the opposite is true in the original dataset.

5.3. Ablation Experiments

We ran an ablation experiment to evaluate how the PLGCN feature aggregation module and the subgraph construction module affected the performance of the recommendations.

5.3.1. Effect of the Feature Aggregation Module

For this section, two variants were designed, and PLGCN was compared to them to verify the performance improvement of the feature aggregation module:

- PLGCN$_{GCN}$: This variant uses the feature aggregation operation in *GCN* [40] to aggregate the user's embedding in both graphs with the following equation:

$$f_{GCN} = \sigma\left(\boldsymbol{W}\left(\boldsymbol{t}_u^{(l)} + \boldsymbol{s}_u^{(l)}\right)\right) \tag{27}$$

- PLGCN$_{GraphSage}$: This variant uses the feature aggregation operation in *GraphSage* [33] to aggregate the user's embedding in both graphs with the following equation:

$$f_{GraphSage} = \sigma\left(\boldsymbol{W}\left(\boldsymbol{t}_u^{(l)} \middle|\middle| \boldsymbol{s}_u^{(l)}\right)\right) \tag{28}$$

In Equations (27) and (28), $\boldsymbol{t}_u^{(l)}$ and $\boldsymbol{s}_u^{(l)}$ denote the embedding of user u propagated through the l-th iteration on the interaction graph and social graph, respectively. $\boldsymbol{W}$ is the

trainable transformation matrix, $||$ means the concatenation operation, and σ is the *tanh* activation function.

As shown in Figure 2, when compared to other models, our proposed feature aggregation module performs the best in all cases. The explanation for the superior performance of PLGCN is that our proposed feature aggregation module first performs a feature transformation on $\boldsymbol{t}_u^{(l+1)}$ and $\boldsymbol{s}_u^{(l+1)}$ and then uses an activation function to activate them nonlinearly, such that a joint space may be created between the user embedding in the two graphs. The multilayer perceptron can be used to explore higher-order feature interactions. In addition, the recommendation performance benefits from the normalization operation, which prevents $\boldsymbol{e}_u^{(l)}$ from increasing with l.

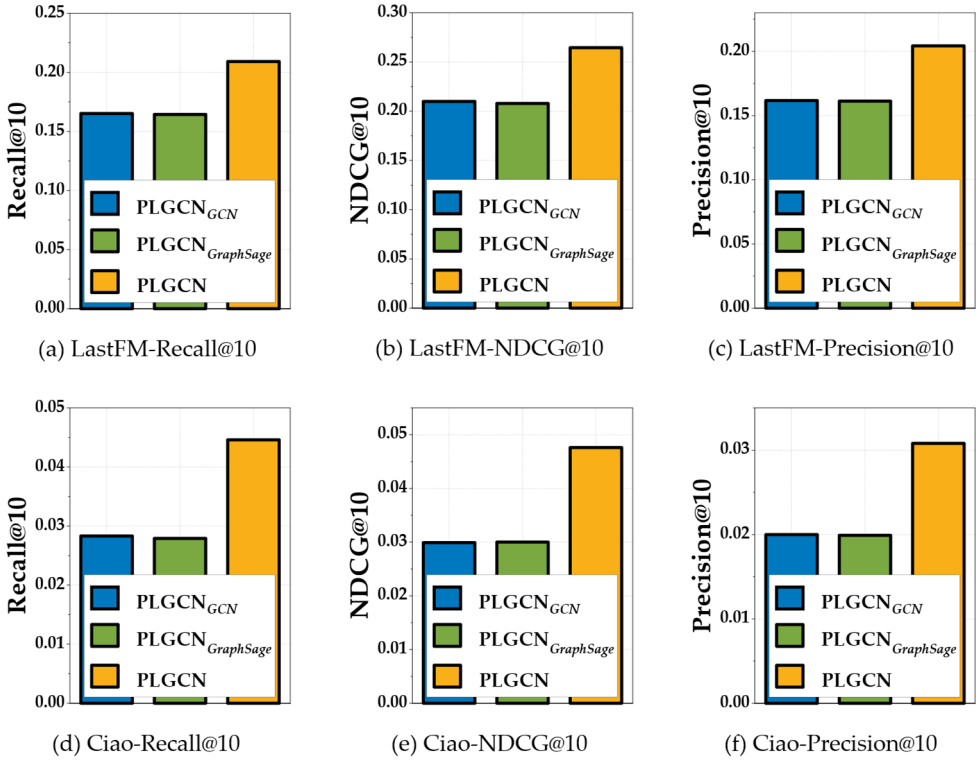

Figure 2. The impact of the feature aggregation module.

5.3.2. Effect of Subgraph Construction Module

This section compares PLGCN with a variant to evaluate whether our proposed subgraph construction module is effective:

- PLGCN$_s$: In this variant, we do not use the subgraph construction module, and we use the same lightweight GNN framework to propagate messages in the social graph and the entire interaction graph.

Table 5 shows the comparison results, which demonstrate that PLGCN has better recommendation performance because the subgraph builder divides users with the same preferences and the items they interact with into a subgraph to filter out the negative information brought by users with different preferences.

Table 5. Performance comparison of PLGCN and its variant on two datasets. The underlined value is the second-best performance, and the bolded value is the best.

Dataset	LastFM			Ciao		
Model	**Precision@10**	**Recall@10**	**NDCG@10**	**Precision@10**	**Recall@10**	**NDCG@10**
$PLGCN_s$	0.2017	0.2065	0.2629	0.0307	0.0441	0.0475
PLGCN	**0.2043**	**0.2091**	**0.2646**	**0.0308**	**0.0446**	**0.0476**

$PLGCN_s$

5.4. Impact Analysis for Hyperparameters

Two crucial parameters affect the performance of PLGCN: the propagation layer number (L) and the subgraph number (N_c). We investigate how they impact the model in this section.

5.4.1. Impact of the Number of Propagation Layers

To explore how the model's performance is impacted by the number L of propagation layers, we kept the other parameters constant and changed L to [1–5]. We display the experimental results in Figure 3, and there is a noticeable improvement in PLGCN's performance when the value of L is increased from 1 to 3 on the original LastFM dataset, and the model performs best at $L = 3$. The original Ciao dataset shows a similar trend, where the model attains the highest performance at $k = 3$ and the performance decreases when k is greater than 3. We inferred from our observations that the recommended performance of the model may be negatively affected by the oversmoothing effect caused by too large a K value, and therefore, we set the K value to 2–4, which is a reasonable choice.

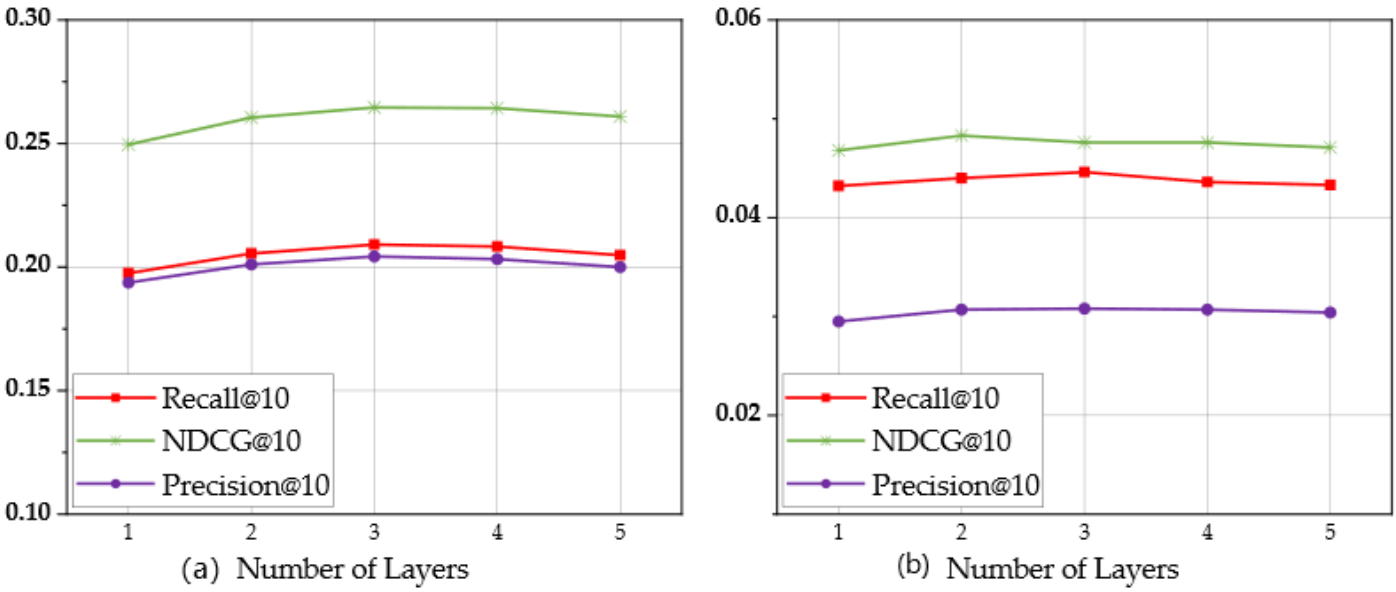

Figure 3. The impact of the number of propagation layers L. (**a**) LastFM dataset and (**b**) Ciao dataset.

5.4.2. Impact of the Number of Subgraphs

The performance of the PLGCN is examined in this section in relation to various subgraph N_c counts. We set N_c to [2–4], and the other parameters were constant. Figure 4 displays the outcomes, where $PLGCN_2$, $PLGCN_3$, and $PLGCN_4$ represent the PLGCN model when N_c is 2, 3, and 4, respectively. It can be seen that $PLGCN_2$ performs best in most cases when there are three propagation layers. It can be assumed that there are fewer layers of propagation at this point, and a node in the subgraph of $PLGCN_2$ has more nodes

connected in a short distance and acquires more information than the nodes in $PLGCN_3$ and $PLGCN_4$, so it performs better.

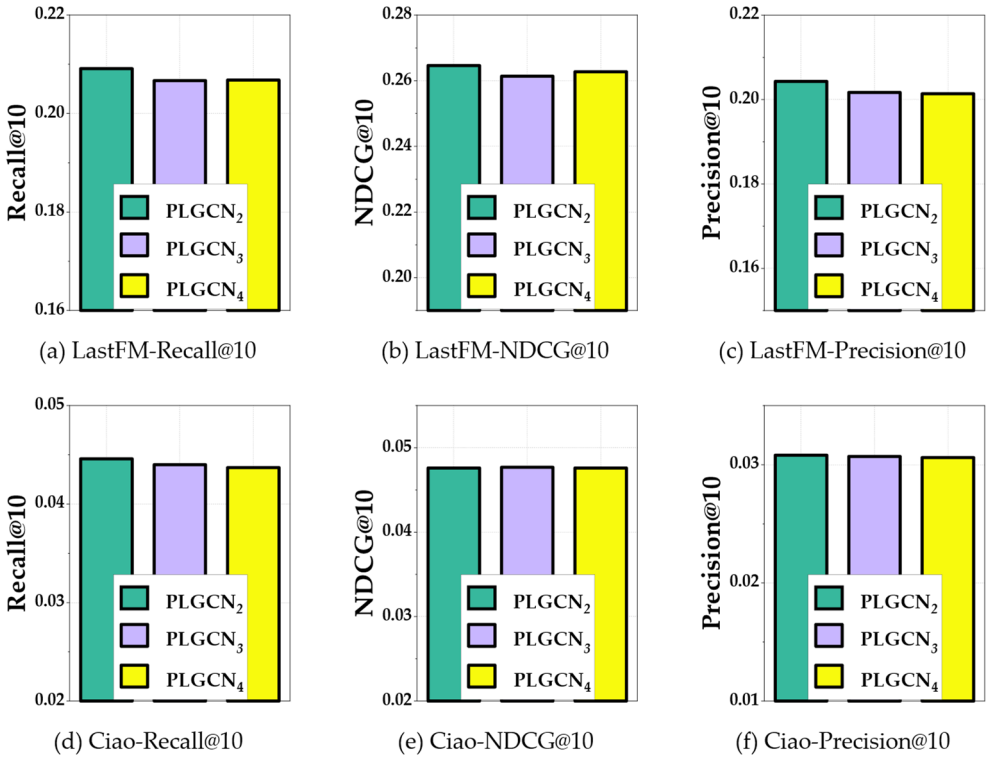

Figure 4. The impact of the number of subgraphs N_c.

6. Discussion

In our experiments, we showed that our graph neural network-based social recommendation model outperforms some previous recommendation models ([8,27,28]). Compared with [28], we find that adding social information to a recommender system does improve the recommendation performance, while compared with [8,27], we find that the quality of social information and the method of using social information also have a crucial impact. In some cases, the performance improvement of our proposed method, PLGCN, is more evident in the cold-start scenario. We believe that cold-start users have fewer interaction data, and negative information has a greater impact on recommendation performance. By filtering out negative information, we substantially improve the recommendation performance.

However, our proposed model also has some limitations, which highlight opportunities for future research. For example, we only consider user preferences in constructing subgraphs, while other social features such as friendship networks or trust levels can be incorporated to enhance the social filtering process. Additionally, our feature aggregation module only includes user embeddings with social and interaction information. It could be extended to include more diverse information sources, such as temporal information or user-generated content.

7. Conclusions

Most of the existing social recommendation models only take higher-order collaborative signals into account, without paying attention to the negative signals in these signals, which negatively affects the models' recommendation performance. We propose the PLGCN, a novel social recommendation model based on GCN, as a solution to this issue, which incorporates unsupervised learning to classify users based on their preferences, allowing for more effective filtering of irrelevant and negative information from high-order neighbor nodes. This enables PLGCN to provide more personalized and accurate recommendations. Moreover, we designed a novel feature aggregation module to

better aggregate user representations in both graphs. We evaluated PLGCN against other SOTA models on two datasets, and the outcomes demonstrated that PLGCN outperforms them. Furthermore, PLGCN adopts a lightweight GNN framework that removes nonlinear activation and feature transformation operations, which mitigates the overfitting issue and enables faster and more efficient training and inference. Our proposed model can be applied to diverse social recommendation scenarios, such as e-commerce, social media, and content recommendation.

However, our model still has limitations. First, it relies on the assumption that social connections effectively capture users' preferences. In reality, users may connect for various reasons, and their social networks may not fully reflect their preferences, which could affect the accuracy of our approach. Second, our experiments were conducted on specific datasets, and the performance of our method may vary on other datasets or domains. Further evaluation is necessary to validate the effectiveness and generalizability of our method. Third, our model assumes a static social network structure and does not consider dynamic changes over time. Future work can explore incorporating dynamic social information to improve the performance of social recommendation methods.

In terms of future work, we plan to investigate several areas for further improvement. First, we would like to explore the use of more complex graph neural network architectures to capture even more nuanced social relationships and better incorporate users' social behavior. Second, we plan to investigate the use of additional data sources, such as user-generated content and location data, to enhance our model's performance and provide more personalized recommendations. Finally, we will explore the use of different datasets and evaluation metrics to better capture the effectiveness of our model and ensure that our recommendations are not only accurate but also diverse and novel.

Author Contributions: Formal analysis, H.X. and L.T.; investigation, G.W.; methodology, H.X. and L.T.; project administration, G.W.; resources, X.J. and E.Z.; data curation, H.X. and L.T.; supervision, L.T.; validation, H.X. and X.J.; writing—original draft preparation, H.X.; writing—review and editing, X.J and L.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Anhui Province Science and Technology Major Special Projects (Project No. 202103b06020013), Anhui Provincial Natural Science Foundation Project (Project No. 2108085MF209), and the Open Fund Project of Anhui Provincial Key Laboratory of Intelligent Agricultural Technology and Equipment (Project No. APKLSATE2021X008).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This research employed publicly available datasets for its experimental studies.

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

References

1. Cialdini, R.B.; Goldstein, N.J. Social influence: Compliance and conformity. *Annu. Rev. Psychol.* **2004**, *55*, 591–621. [CrossRef] [PubMed]
2. McPherson, M.; Smith-Lovin, L.; Cook, J.M. Birds of a feather: Homophily in social networks. *Annu. Rev. Sociol.* **2001**, *27*, 415–444. [CrossRef]
3. Knoke, D.; Yang, S. *Social Network Analysis*; SAGE publications: London, UK, 2019.
4. Ma, H.; Zhou, D.; Liu, C.; Lyu, M.R.; King, I. Recommender systems with social regularization. In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, Hong Kong, China, 9–12 February 2011; pp. 287–296.
5. Tang, J.; Wang, S.; Hu, X.; Yin, D.; Bi, Y.; Chang, Y.; Liu, H. Recommendation with social dimensions. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
6. Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; Yin, D. Graph neural networks for social recommendation. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 417–426.
7. Wu, L.; Li, J.; Sun, P.; Hong, R.; Ge, Y.; Wang, M. Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 4753–4766. [CrossRef]

8. Wu, L.; Sun, P.; Fu, Y.; Hong, R.; Wang, X.; Wang, M. A neural influence diffusion model for social recommendation. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, Franch, 21–25 July 2019; pp. 235–244.
9. Fout, A.; Byrd, J.; Shariat, B.; Ben-Hur, A. Protein interface prediction using graph convolutional networks. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6533–6542.
10. Duvenaud, D.K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R.P. Convolutional networks on graphs for learning molecular fingerprints. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 2224–2232.
11. Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V.; Riley, P. Molecular graph convolutions: Moving beyond fingerprints. *J. Comput. -Aided Mol. Des.* **2016**, *30*, 595–608. [CrossRef] [PubMed]
12. Wu, S.; Sun, F.; Zhang, W.; Xie, X.; Cui, B. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–37. [CrossRef]
13. Eksombatchai, C.; Jindal, P.; Liu, J.Z.; Liu, Y.; Sharma, R.; Sugnet, C.; Ulrich, M.; Leskovec, J. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 1775–1784.
14. Wu, Q.; Zhang, H.; Gao, X.; He, P.; Weng, P.; Gao, H.; Chen, G. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 2091–2102.
15. Chen, T.; Wong RC, W. An efficient and effective framework for session-based social recommendation. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Online, 8–12 March 2021; pp. 400–408.
16. Liu, F.; Cheng, Z.; Zhu, L.; Gao, Z.; Nie, L. Interest-aware message-passing gcn for recommendation. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 1296–1305.
17. Wang, X.; He, X.; Nie, L.; Chua, T.S. Item silk road: Recommending items from information domains to social users. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 185–194.
18. Lin, T.H.; Gao, C.; Li, Y. Recommender systems with characterized social regularization. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 1767–1770.
19. Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 135–142.
20. Zhao, T.; McAuley, J.; King, I. Leveraging social connections to improve personalized ranking for collaborative filtering. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, Shanghai, China, 3–7 November 2014; pp. 261–270.
21. Yu, J.; Gao, M.; Li, J.; Yin, H.; Liu, H. Adaptive implicit friends identification over heterogeneous network for social recommendation. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Turin, Italy, 22–26 October 2018; pp. 357–366.
22. Guo, G.; Zhang, J.; Yorke-Smith, N. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In Proceedings of the AAAI Conference on Artificial Intelligence, Chicago, Il, USA, 25–30 January 2015; Volume 29.
23. Chaney AJ, B.; Blei, D.M.; Eliassi-Rad, T. A probabilistic model for using social networks in personalized item recommendation. In Proceedings of the 9th ACM Conference on Recommender Systems, Vienna, Austria, 16–20 September 2015; pp. 43–50.
24. Ma, H.; King, I.; Lyu, M.R. Learning to recommend with social trust ensemble. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Boston, MA, USA, 19–23 July 2009; pp. 203–210.
25. Koren, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Vegas, NV, USA, 22 February 2008; pp. 426–434.
26. Liu, Y.; Chen, L.; He, X.; Peng, J.; Zheng, Z.; Tang, J. Modelling high-order social relations for item recommendation. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 4385–4397. [CrossRef]
27. Liao, J.; Zhou, W.; Luo, F.; Wen, J.; Gao, M.; Li, X.; Zeng, J. SocialLGN: Light graph convolution network for social recommendation. *Inf. Sci.* **2022**, *589*, 595–607. [CrossRef]
28. 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, Online, 25–30 July 2020; pp. 639–648.
29. Hu, Y.; Zhan, P.; Xu, Y.; Zhao, J.; Li, Y.; Li, X. Temporal representation learning for time series classification. *Neural Comput. Appl.* **2021**, *33*, 3169–3182. [CrossRef]
30. Hartigan, J.A.; Wong, M.A. Algorithm AS 136: A k-means clustering algorithm. *J. R. Stat. Society. Ser. C (Appl. Stat.)* **1979**, *28*, 100–108. [CrossRef]
31. Yang, J.; Zhang, D.; Frangi, A.F.; Yang, J.Y. Two-dimensional PCA: A new approach to appearance-based face representation and recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 131–137. [CrossRef] [PubMed]
32. Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T. Simplifying graph convolutional networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 10–15 June 2019; pp. 6861–6871.

33. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1025–1035.
34. Cantador, I.; Brusilovsky, P.; Kuflik, T. Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011). In Proceedings of the Fifth ACM Conference on Recommender Systems, Chicago, IL, USA, 14 October 2011; pp. 387–388.
35. Tang, J.; Gao, H.; Liu, H. mTrust: Discerning multi-faceted trust in a connected world. In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, Washington, DC, USA, 8–12 February 2012; pp. 93–102.
36. Xu, H.; Huang, C.; Xu, Y.; Xia, L.; Xing, H.; Yin, D. Global context enhanced social recommendation with hierarchical graph neural networks. In Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; IEEE: Piscataway, NJ, USA; pp. 701–710.
37. Lin, J.; Chen, S.; Wang, J. Graph neural networks with dynamic and static representations for social recommendation. In Proceedings of the Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, 11–14 April 2022; Springer International Publishing: Cham, Switzerland, 2022; pp. 264–271.
38. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. *arXiv* **2012**, arXiv:1205.2618.
39. Wang, X.; He, X.; Wang, M.; Feng, F.; Chua, T.S. Neural graph collaborative filtering. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 165–174.
40. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.