Complex & Intelligent Systems (2023) 9:3819–3833
https://doi.org/10.1007/s40747-022-00849-9

ORIGINAL ARTICLE

Unexpected interest recommender system with graph neural network

Hongbin Xia[1,2] · **Kai Huang**[1] · **Yuan Liu**[1,2]

Received: 22 April 2022 / Accepted: 31 July 2022 / Published online: 22 August 2022

Abstract
Traditional recommender systems often face the filter bubble problem when they focus on recommending familiar items to users. The over-specialized recommended contents will make users bored. To solve this problem, researchers have proposed models that focus on unexpectedness, but these models all suffer from incomplete learning of features. To address this problem, we propose an unexpected interest recommender system with graph neural network (UIRS-GNN). First, we preprocess the input data with a graph convolutional network. It enriches user and item feature vectors by aggregating neighborhood information. Second, we transform the GRU and propose the attention-based long short-term gated recurrent unit network to learn user preferences hidden in historical behavior sequences. Then, we input the preprocessed feature vectors of users and items into the unexpected interest model, and solve the problem of insufficient feature information learning by aggregating neighborhood information. Furthermore, our model also alleviates data sparsity due to our deep learning feature information. Finally, empirical evaluations with several competitive baseline models on three real-world datasets reveal the superior performance of UIRS-GNN.

Keywords Graph neural network · Long term and short term · Recommender system · Unexpectedness

Introduction

The massive volume of user–item interactions' data on the internet today has expedited the creation of diverse personalized recommendation models with the goal of presenting to users a set of unseen items that may be of interest to them. Among them, content-based recommendation [1] and collaborative filtering recommendation [2] are two representative methods. To learn the interests of users' more deeply, session-based recommendation [3] has also been incorporated to learn information such as user behavior sequences, and has achieved success. However, recent studies have shown that traditional recommendation methods often lead to over-professional recommended content [4], which leads to user boredom [5], and even reduce user satisfaction with the product.

To address these issues, researchers propose to incorporate an unexpected measure of user interest in the recommendation models [6]. These models recommend novel, unexpected and satisfying content to users. These contents are not completely in line with or deviate from the user's interests, but are reasonable recommendations made after deep learning of the users' characteristics and behavior sequences. Unlike traditional diversity [7] methods that focus on the differences between recommended items, the unexpected measure detects the deviation between user interests and recommended items. Through a series of studies on unexpected measurement algorithms [6, 8, 9], it is found that the model with unexpected measurement will recommend more satisfying items to users.

However, because researchers pay more attention to optimizing the indicators of unexpected metrics, they lack deeper learning of users and items. This will cause users to be dissatisfied with the recommended content. The problem is that the model does not deep learn the potential characteristics of users and items while learning the deviation between user interests and recommended items. This will cause a certain degree of misinterpretation of the range of users' interest preferences, and result in the recommended content being too relevant or too unexpected. For example, each user's

✉ Hongbin Xia
hbxia@jiangnan.edu.cn

[1] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, People's Republic of China

[2] Jiangsu Key Laboratory of Media Design and Software Technology, Wuxi, People's Republic of China

needs for recommended content are personalized. Some users prefer things they are familiar with, while others are more willing to accept novel things. They also have different definitions of whether the item recommended to them is familiar or surprising. It is necessary to focus on the different interest preferences and personal characteristics of each user, and customize the recommended content that meets their relevance and unexpectedness.

In this paper, we propose an unexpected interest recommender system with graph neural network (UIRS-GNN) to address the current limitations of models. Building on the work of the PURS model [9], we use a graph neural network to aggregate the features of neighbor nodes into the target node. Then we use the attention-based [10] long short-term gated recurrent unit network (A-LSGRU) to model the user's behavior sequence, and respectively, learn the user's long-term preference and short-term preference. A-LSGRU will capture each user's personalized content of interest. In the next, we input the richer feature target nodes in the unexpectedness model and model the unexpectedness metric as the weighted distance between the user's interest and the recommended item. At last, we combine the A-LSGRU and the unexpected interest model to construct a new unexpected interest model.

In summary, the following are our major contributions:

1. We propose the UIRS-GNN, a novel unexpected interest recommendation model which use graph neural network to construct the neighborhood of target node, and aggregate the neighbor node features into the target node. Our model can enrich the feature information of the target node and also improve the feature expression ability.
2. The proposed model can learn the user's interest preference through using the attention-based long short-term gated recurrent unit network (A-LSGRU). We model the user's global and local interest preference to obtain more comprehensive user characteristics.
3. The proposed model is based on user interest preferences, and considers both the relevance and unexpectedness of the recommendation in an end-to-end manner. We can optimize one module independently without affecting the results of the other.
4. We conduct empirical evaluations with several competitive baseline models on three real-world datasets to demonstrate the superior performance of UIRS-GNN.

The remaining parts of this paper are organized in this method. Relation works are discussed in "Related works". In "Unexpected interest recommender system with graph neural network (UIRS-GNN)", we give the structure and details of the UIRS-GNN model. In "Empirical study", we also provide our model's empirical setup. The result and analysis are described in "Result". In "Conclusions", we summarize our work and what we can do in the future.

Related works

This section reviews related recommender systems techniques, which include three part: the recommendation with graph convolutional neural networks, the session-based recommendation focusing on user's behavior sequences, and unexpected interest recommendation.

The recommendation with graph convolutional neural (GCN) networks

Convolutional neural networks have achieved success in different domains such as image [11] and text [12]. In contrast to regular images and text, researchers have begun to generalize convolutions to inherently irregular graphs [13]. Graph convolutional networks have attracted much attention of researchers due to their rigorous theory and relatively efficient performance [14]. The core idea of GCNs is to model message passing or information diffusion in a graph structure to generate node embeddings. Each node obtains its own embedding by aggregating the information of its neighbors, and the messages from the neighbors come from the neighbors of their neighbors, and so on. These models are called convolutions, because the operation of aggregating from neighbors is similar to convolutional layers in computer vision. GraphSAGE [15] expands GCN into an inductive learning task by training a function that aggregates the neighbors of nodes (convolutional layer), which generalizes to unknown nodes. Following the success of applying GCNs to graphs, researchers propose to learn latent features of users and items by passing information on a user–item interaction graph under the graph [16]· Among them, PinSage [17] uses a combination of random walks and graph convolutions to capture the features of the graph structure and the features of nodes to generate embedded representations of nodes; NGCF [18] explicitly models user–item to effectively inject collaborative signals into the embedding process; LightGCN [19] simplifies the learning process by deleting the feature transformation and nonlinear activation operations of traditional GCNs, and proves that these two operations are effective in recommender systems with no significant effect. These attempts to apply GCNs to recommender systems simply transform the user–item interaction matrix into a graph and focus on the relevance of recommendations. Compared to these models, we use GCN as a way of data preprocessing. We take the data processed by neighborhood aggregation as the input of the A-LSGRU and the unexpected interest model. These data will then be processed to discover the user's preference interest.

The session-based recommendation focusing on user's behavior sequences

Traditional CF methods such as matrix factorization fail in session-based recommendation because user profiles cannot be constructed from past user behaviors. A natural solution to this problem is the item-to-item recommendation method [20]. The model will precompute an item-to-item similarity matrix from the available session data, and consider the items which frequently clicked in the session similar. These similarities are used to create user interest profiles. The method, although simple, has been shown to be effective and then widely used. However, these methods only consider the user's last click and effectively ignoring information about previous clicks. It is necessary to completely model the user's behavior sequence to learn the user's characteristics. Researchers have found that RNNs are very effective when dealing with sequence data [21]. RNNs have been applied to image, video captioning, time series prediction, natural language processing, etc. long short-term memory network (LSTM) [22] and gated recurrent unit network (GRU) [23] are two variants of RNN. They are relative to RNN by introducing a gating mechanism to control the accumulation speed of information, including selective of adding new information and selectively forgetting previously accumulated information. This helps to improve the long-range dependency problem of RNN and deep learn the user's behavior sequence. However, different users have different preferences for the same recommendation and even the same user has different preferences for similar recommended items in different sessions. It is necessary to model a personalized session recommender system to learn user's behavior sequences. DIN [24], DeepFM [25], Wide and Deep [26], PNN [27] recommend personalized content for each user through modeling the features of users and items, and the user's behavioral interest sequence. Compared with these conversational recommender systems, we introduce an attention mechanism to capture the interest bias of each user. It assigns different weights to users according to user's behavior sequences. In addition, we emphasize the weighting of short-term interests. We separately extract the last interaction in the user behavior sequence as the user's short-term interest. Then we spliced it with the user's long-term interest as the user's feature.

Unexpected interest recommendation

To address the problems of over-specialized recommendations and user boredom, researchers have proposed the concept of unexpectedness. The unexpectedness measures users' emotional responses to the item they did not know before, and detects the surprise of target users to broaden user's interest preference and improve the user's satisfaction

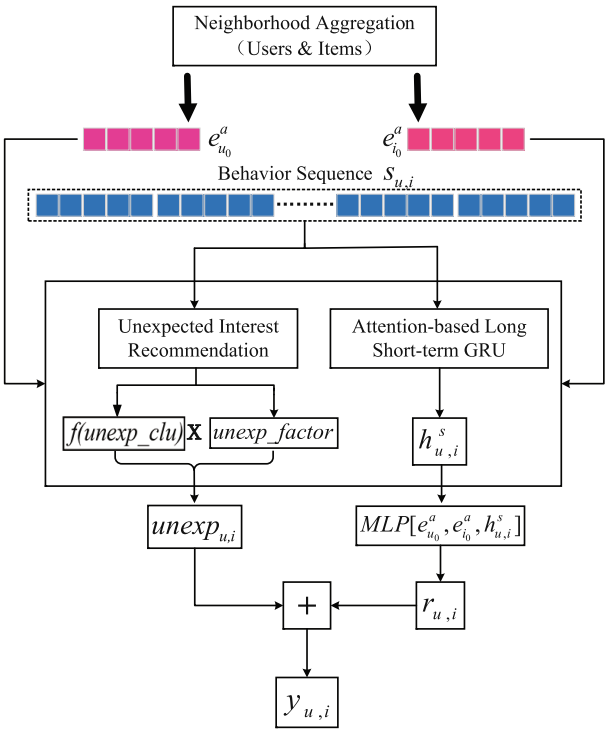


Fig. 1 Framework of UIRS-GNN

[28]. Unlike evaluation criteria such as diversity, unexpectedness measures those recommendations that are not included in the user's previous purchases or deviate from the user's expectations. It is usually defined as the distance between the target item in the feature space and the user's interest set. But as pointed out in the literature [16], it is simpler to compute the distance of item embeddings in the latent space than in the feature space. Auralist [29] improves user satisfaction by balancing accuracy and novelty measures while using topic modeling; PURS [9] provides multi-cluster modeling of user interests in the latent space, as well as through self-attention mechanisms and selecting appropriate Unexpected activation functions to achieve personalized unanticipated recommendations. These models generally suffer from insufficient feature learning. Therefore, we introduce GCN to aggregate the features of neighbor nodes into the target node. It greatly enriches the features of users and items and effectively alleviating the problem of insufficient feature learning.

Unexpected interest recommender system with graph neural network (UIRS-GNN)

The structure of UIRS-GNN model is shown in Fig. 1. It consists of three parts: neighborhood aggregation with graph neural network, the attention-based long short-term gated

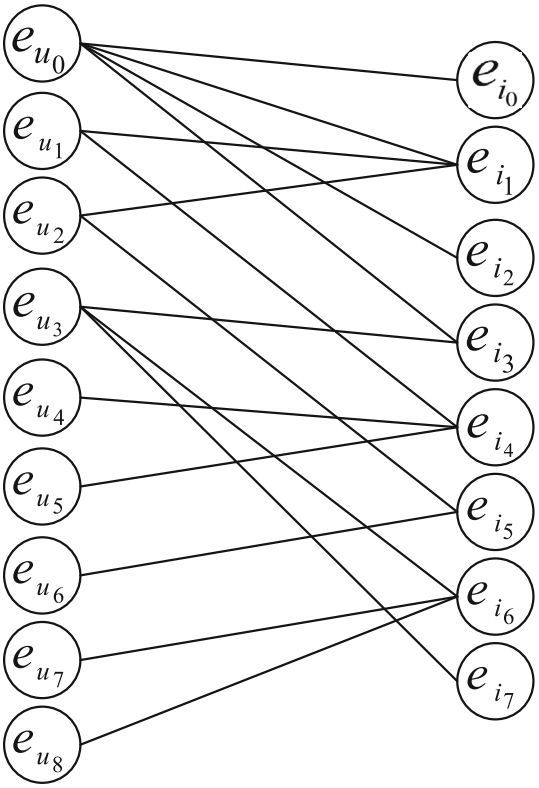


Fig. 2 Graph of user–item interaction

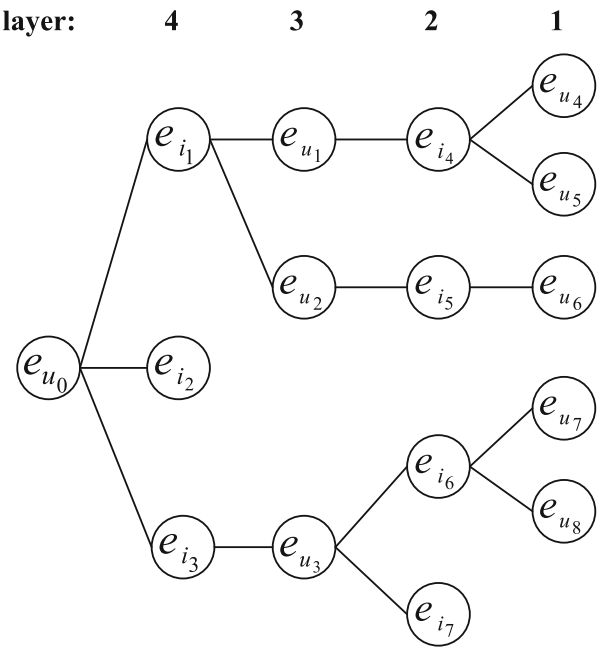


Fig. 3 Graph of user–item adjacency

recurrent unit network (A-LSGRU), and unexpected interest recommendation. We will introduce them as following.

Neighborhood aggregation with graph neural network

Neighborhood aggregation

Neighborhood aggregation is the initial step of our model. The main function of this step is to aggregate the features of neighbor nodes into the target node, so as to enrich the features of the target node, and provide input data for the A-LSGRU and the unexpected interest model.

In the initial steps, we need to associate users and items with their embedded ID. Here, we set $u_i \in U$ to represent users, where U represents the total set of users; $i_i \in I$ items, where I represents the total set of items. We use e_u to represent the user's embedding and e_i to represent the item's embedding, and then use the user–item adjacency graph to learn latent features and propagate the learned features to the next layer. The corresponding interaction graph and adjacency graph are shown in Figs. 2 and 3.

After establishing the adjacency relationship between users and items, we need to aggregate the feature information of these neighbor nodes into the target node. The propagation formula [18] is as follows:

$$\begin{cases} e_u^{k+1} = \sigma\left(w_1 e_u^k + \sum\limits_{i \in N_u} \frac{1}{\sqrt{|N_u||N_i|}}\left(w_1 e_i^k + w_2\left(e_i^k \odot e_u^k\right)\right)\right) \\ e_i^{k+1} = \sigma\left(w_1 e_i^k + \sum\limits_{i \in N_i} \frac{1}{\sqrt{|N_u||N_i|}}\left(w_1 e_u^k + w_2\left(e_u^k \odot e_i^k\right)\right)\right), \end{cases} \tag{1}$$

where e_u^k and e_i^k represent the embedding vectors of user u and item u in the layer, σ represent the nonlinear activation function, w_1 and w_2 represent the weight matrix for feature transformation at each layer, N_u and N_i represent the adjacent nodes of user u and item i.

After the model obtains the node embedding vector and adjacency information, it will follow the order of the layers in Fig. 3, starting from the first layer to obtain the description vector of each layer about the user $\left(e_u^1, e_u^2 \ldots e_u^k\right)$ and the description vector of the item $\left(e_i^1, e_i^2 \ldots e_i^k\right)$, and then use these obtained embedding vectors. Connect with the embedding vector of the target node to obtain the final user e_u^a and item e_i^a embedding sum, which a represents the neighborhood aggregation operation, generally using the splicing || operation. The formula [18] is as follows:

$$\begin{cases} e_u^a = e_u^1 \| e_u^2 \| \cdots \| e_u^k \\ e_i^a = e_i^1 \| e_i^2 \| \cdots \| e_i^k \end{cases}. \tag{2}$$

LightGCN

We use a binary data set without textual information, and choose a simplified GCN model-LightGCN, which greatly simplifies the operation of neighborhood aggregation.

The idea of neighborhood aggregation in the field of recommender systems comes from the traditional GCN—the relevant knowledge of graph convolutional neural networks. The core is the neighborhood aggregation function, such as formula (1). The purpose is to aggregate the target node and the neighbor nodes of the Kth layer as a feature representation. It includes two essential operations—nonlinear activation function and feature transformation, which have a pivotal role in the task of dealing with nodes with rich semantics. But in recommendation tasks where only user and item ids are input, they may not be effective. He et al. [19] proposed LightGCN, which removes feature transformation and nonlinear activation function according to the characteristics of sparse recommendation task node information and low feature dimension. LightGCN can improve the training speed and accuracy.

The improvement of LightGCN is mainly in the reasonable deletion of nonlinear activation function and feature transformation. First, we need to perform a neighborhood aggregation operation. The aggregation formula [19] of LightGCN is as follows:

$$\begin{cases} e_u^{k+1} = \sum\limits_{i \in N_u} \dfrac{1}{\sqrt{|N_u||N_i|}} e_i^k \\ e_i^{k+1} = \sum\limits_{u \in N_i} \dfrac{1}{\sqrt{|N_i||N_u|}} e_u^k \end{cases}, \tag{3}$$

where e_u^k and e_i^k represent the embedding vectors of user u and item i at k-layer, N_u and N_i represent the adjacent nodes of user u and item i. Looking at formula (3), the most obvious feature is that the nonlinear activation function and feature transformation in formula (1) are deleted. In addition, the formula cancels the self-connection operation. LightGCN has captured the information of the target node in the operation of layer combination, so the self-connection is deleted to avoid redundant operations.

After obtaining the node feature information of each layer, the model uses the weighted method to fuse the target node and the neighbor nodes. The formula [19] is as follows:

$$\begin{cases} e_u^a = \sum\limits_{k=1}^{K} a_k e_u^k \\ e_i^a = \sum\limits_{k=1}^{K} a_k e_i^k \end{cases}, \tag{4}$$

where K denotes the number of neighborhood layers, a_k denotes the weight of the Kth layer embedding, here we use a simple $1/(K+1)$ to denote it, which has been proven to work well.

Attention-based long short-term gated recurrent unit network (A-LSGRU)

Our model uses an attention-based long short-term gated recurrent unit network to enrich user features. The purpose is to learn the user's long-term and short-term behavior sequences that change over time and discover the user's hidden preferences and interests in the behavior sequence.

In recommender systems, for an item i, our purpose is to predict whether user u will click on the item, and this prediction depends to some extent on whether the user's interest preference matches the item. In this section, we will use an attention mechanism based long short-term memory neural network to learn the user's preference interests. We will use $s_i \in I_u$ to represent each node in the behavior sequence, set the behavior sequence to $[s_1, s_2, s_3 \ldots, s_n]$, sorted by timestamp, which I_u represents the user's behavior sequence (click item sequence), model as shown in Fig. 4.

Node processing

In our model, we convert user's behavior sequence into an embedding vector. It is worth noting that the node information at this time is not just the embedding vector containing its own information, but the nodes aggregated by the graph neural network. Each node in the behavior sequence contains the feature information of its neighborhood, which greatly enriches the node features. The node at this time should be represented as $s_i^a \in I_u$, where a represents the domain aggregation operation, but for the sake of brevity, it is still used s_i.

Gated recurrent unit network

When dealing with sequence information, RNN [30] has unique advantages. It is a kind of neural network with short-term memory, which can not only receive information from other neurons, but also receive its own information. We use a gated recurrent unit network GRU to model user interest and capture temporal and click information in behavior sequences. Compared with other recurrent neural networks such as traditional RNN and LSTM, GRU is computationally more compact and efficient.

First, we transform the user's behavior sequences into corresponding embedding vectors in the feature space, which are then fed into the GRU. The update function [23] of its

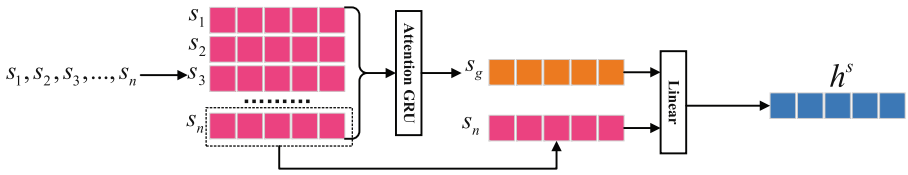


Fig. 4 Attention-based long short-term gated recurrent unit network (A-LSGRU)

learning process is as follows:

$$Z_t = \sigma(W_Z x_t + U_z h_{t-1} + b_Z) \tag{5}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{6}$$

$$h_t = Z_t \odot h_{t-1} + (1 - Z_t) \odot \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h), \tag{7}$$

where Z_t and r_t represent the update gate and reset gate of the GRU, respectively, W_Z and W_r, U_Z and U_r are the weight matrix of the current input x_t and the last state h_{t-1} of the update gate and the reset gate. b_Z and b_r are the bias vectors, and σ represent the sigmoid function. W_h and b_h are the weight matrix and bias vector of the candidate state, and $\odot$ represent the Hadamard product. After the user behavior sequence embedding vector is input into the GRU, the update gate such as Eq. (5) and the reset gate such as Eq. (6) will control how much information the current state needs to retain from the historical information and how much new information is accepted from the candidate state. Then, the GRU will obtain the final state h_t according to the combined algorithm of the current state, candidate state, update gate and reset gate.

However, while learning the user behavior sequence, we found that each node has different correlations to user preference interests. For example, when processing a behavior sequence of a user whose preference is science fiction movies, the movies related to science fiction in the sequence are more likely to satisfy the user, so its weight value should be higher. On the contrary, other types of movies should appropriately reduce the weight. To capture the user's interest bias, we introduce an attention mechanism when dealing with sequence modeling:

$$u_{t,i} = \sigma(W_3 h_t + W_4 x_i + b_u) \tag{8}$$

$$a_{u,t} = \frac{\exp(u_{t,i})}{\sum_{i=1}^{n} \exp(u_{t,i})} \tag{9}$$

$$S_g = \sum_{i=1}^{n} a_{t,i} x_i, \tag{10}$$

where $u_{t,i}$ denotes the compatible function value of each input node in the sequence with the final state, W_3, W_4 and b_u are the weight matrix and bias vector of formula (8). Then, they are brought into the attention formula (9) to obtain the attention weights, and the global vector is obtained through the weighting function S_g.

Finally, after deriving the global vector, we pay more attention to the user's latest preference interest. We separately extract the last embedding vector in the user behavior sequence as a local preference vector. Then, we concatenate it with the global preference vector and perform a linear transformation to obtain the final sequence mixed embedding $h^s_{u,i}$:

$$h^s_{u,i} = W_5(S_g \| s_n) + b_s, \tag{11}$$

where $W_5 \in R^{d \times 2d}$ is the transformation matrix that compresses the concatenation of two vectors into $R^{d \times d}$ space, b_s is the bias vector of the formula, || is the concatenation operation.

Unexpected interest recommendation

The purpose of our model using the unexpected interest recommendation model is to address filter bubble problem. After learning the characteristics of users, the model will recommend surprisingly content to users.

Currently, recommender systems focus on accurately recommending items related to user interests. However, too much attention to accurate recommendation is likely to lead to a single recommendation item, which will lead to user boredom. Therefore, unexpected interest recommendation began to enter the researchers' perspective. It considers that recommended items should be related to user interests and avoid homogenization. Reza et al. generalized it as serendipity [6]. Our unexpected interest model (as shown in Fig. 5) refers to the paper [9]. We define the method of measuring the unexpectedness as the distance between the recommended item and the user's behavioral interest sequence. However, since the distance function is difficult to define in the feature space, and related methods are also difficult to achieve the best performance, we model the unexpected function in the latent space. It enables the model not only to guarantee the recommendation accuracy, but also to improve the recommendation.

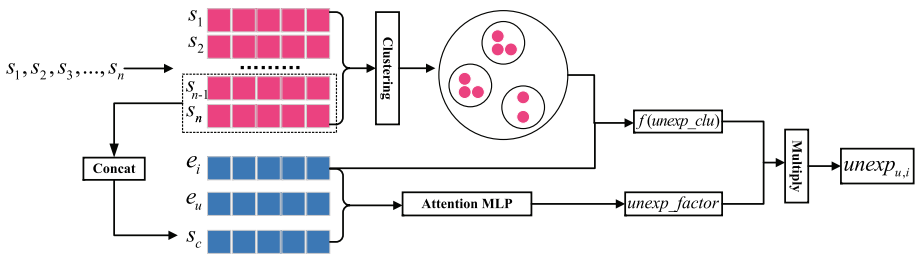
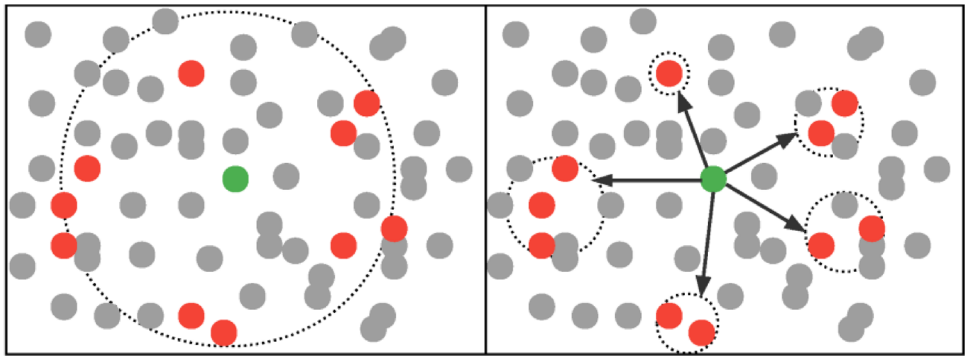


Fig. 5 Model of unexpected interests



(a) Single interest model (b) Cluster interest model

Fig. 6 Interest modeling in latent space

Unexpected function

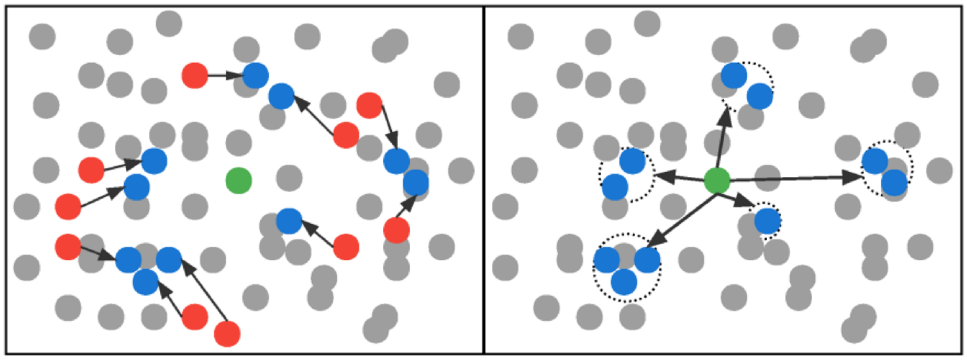
First, we use clustering to model the user's interest space. Compared with classifying all user interests into the same space, clustering can divide the interests of users into different groups according to the similarity. We can exclude some items in the latent space that are not related to user interests and more easily identify the types of user interests in each cluster group. Then, model will learn for different interest types of users and deeply discover the unexpected interests of users. The principle is shown in Fig. 6, where the gray points represent irrelevant items, the red points represent user behavior items, and the green points represent target items.

It is worth noting that the items related to the target user in the latent space are also the items that have been aggregated through the graph neural network. Unlike items that only contain their own information, they contain relevant features of the neighborhood. In the latent space, they are equivalent to a collection of nodes. Visually, their relative positions have changed. As shown in Fig. 7, the red point represents the previous item of the aggregation operation, the blue point represents the post-aggregation operation item. The arrows between the two represent the position movement before and after the aggregation operation, and the green point represents the target item. The model can more accurately model the user's interest clustering after the aggregation operation.

When modeling interest clustering, the choice of clustering algorithm also affects the group of interests. Here, we choose the mean shift algorithm, because it is an unsupervised clustering algorithm. We do not need to choose the number and shape of clusters, and can flexibly implement different modeling for each user. We set the user's behavior sequence as $[s_1, s_2, \ldots, s_n]$, and the embedding mapped in the latent space as $[l_1, l_2, \ldots, l_n]$. Then, we use the Mean Shift algorithm to cluster the embeddings to obtain user interest clusters $[C_1, C_2, \ldots, C_n]$. Referring to the method of Panagiotis et al.[8], we model the unexpected function as the weighted average distance between the target item and each cluster. The formula [8] is as follows:

$$\text{unexp\_clu}_{u,i} = \sum_{k=1}^{N} d(e_i, C_k) \times \frac{|C_k|}{\sum_{k=1}^{N} |C_k|}. \tag{12}$$

After the unexpected result value is obtained, it can be used directly for the scoring function. But considering that the clustering operation is invisible, the obtained unexpected value is likely to deviate excessively. These values may cause the model to tend to recommend items with a high degree of surprise, which is likely to have a negative impact. Therefore, to make the recommendation within a certain controllable range, we need to weigh the unexpected values. Panagiotis et al. [8] recommended using a unimodal function to adjust unexpected values, which needs to satisfy the four necessary conditions of continuity, boundedness, unimodality, and short-tail. We choose a commonly used function $f(x) = xe^{-x}$ in the gamma function [31] as the activation function. This activation function satisfies all the conditions, and is enough simple and effective.



(a) Item cluster migration (b) Cluster interest model

Fig. 7 The influence of neighborhood aggregation on cluster interest model

Unexpected factor

As mentioned in the previous section, we should use a unimodal activation function to keep the recommendation within the controllable range. However, this function focuses on improving the unexpectedness of the recommendation, and we also need to deal with the relevance of the recommendation.

Each user's interest preferences are different, which is also reflected in the unexpectedness. According to the method of Li et al. [9], we set an unexpected factor to adjust the user's personal unexpected interest preference. First, we set the behavior sequence of user u as $[s_1, s_2, \ldots, s_n]$. To obtain the user's latest interest preference, we choosing the most recent behavior as the personalization factor for each user instead of using the entire sequence to learn the user's interest preference. This behavior length is a hyperparameter that can be adjusted manually. We set the last three items as personalized windows. Similarly, since the embeddings in the personalized window are not unique, their influence on the target user is also different. To capture the differential influence of different items on the target user, we use an attention mechanism to learn their influence weights. Then, we use the multi-layer perceptron to integrate the output results. The formula [9] is as follows:

$$\text{unexp\_factor}_{u,i} = MLP\left(e_u, \sum_{k=1}^{K} a_{k,i} s_k, e_i\right), \tag{13}$$

where MLP is a multi-layer perceptron, and a is the attention factor of each item to the target user.

After the model finds the formulas representing unexpectedness and correlation respectively, we multiply the two results to obtain the final unexpected function result. The formula [9] is as follows:

$$\text{unexp}_{u,i} = f\left(\text{unexp\_clu}_{u,i}\right) \times \text{unexp\_factor}_{u,i}. \tag{14}$$

Model training

After the model receives all the variables it needs, we need to integrate them to calculate the score $Z^{\sim}_{u,i}$. First, we put the target user embedding e_u, the target user embedding e_i, and the sequence hybrid embedding $h^s_{u,i}$ into the multi-layer perceptron MLP network to get the relevance score $r_{u,i}$. Then, we add it to the unexpected score $\text{unexp}_{u,i}$ to get the final score. The formula is as follows:

$$r_{u,i} = MLP\left[e^a_{u_0}, e^a_{i_0}, h^s_{u,i}\right] \tag{15}$$

$$Z^{\sim}_{u,i} = r_{u,i} + \text{unexp}_{u,i}. \tag{16}$$

Second, we apply the sigmoid function to get the output vector of the model $y^{\sim}$:

$$y^{\sim} = sigmoid(Z^{\sim}), \tag{17}$$

where $Z^{\sim}$ denotes the recommendation score for all candidate items, $y^{\sim}$ denotes the probability that the node becomes the next target.

Finally, we define the loss function as the cross-entropy of the predicted result $y^{\sim}$ and the true value y, and its formula is as follows:

$$loss = \sum_{i=1}^{n} -\left[y_i \ln\left(y_i^{\sim}\right) + (1 - y_i)\ln\left(1 - y_i^{\sim}\right)\right]. \tag{18}$$

The workflow of UIRS-GNN is depicted in Algorithm 1.

Algorithm 1 Training Process for UIRS-GNN

Input: User embedding $e_u \in U$, Item embedding $e_i \in I$, User-Item interaction matrix, User's behavior sequence $s \in I_u$

Output: Recommendation lists $y^{\sim} \in R^{UxI}$

while *stopping criteria not satisfied* **do**
  for *each user u and item i* **do**
    for *layer 1~k* **do**
      $e_u^{k+1} = AGG(e_i^k, \{e_i^k : i \in N_u\})$
      $e_i^{k+1} = AGG(e_u^k, \{e_u^k : u \in N_i\})$
    end for
    then
      $e_u = \sum_{k=1}^{K} a_k e_u^k$
      $e_i = \sum_{k=1}^{K} a_k e_i^k$
  end for
  for *each user u* **do**
    Learning user preference interests through behavior sequences
    $h_t = GRU(u, s_u)$
    $S_g = Attention(h_t)$
    $h_{u,i}^s = Concat[S_g \| s_n]$
    $r_{u,i} = MLP[e_u, e_i, h_{u,i}^s]$
    Learn unexpected interests of users
    $unexp\_clu_{u,i} = MeanShift(e_i, C)$
    $unexp\_factor_{u,i} = MLP(e_u, \sum_{k=1}^{K} a_{k,i} s_k, e_i)$
    $unexp_{u,i} = multiply(unexp\_clu_{u,i}, unexp\_factor_{u,i})$
  end for
  $y^{\sim} = sigmoid(r_{u,i} + unexp_{u,i})$
end while

Empirical study

Dataset

We validate our model on three real datasets: Yelp Challenge Dataset,[1] which contains information about users, restaurants, and user ratings of restaurants; MovieLens 1M[2] and MovieLens 10M,[3] which includes users, movies, and user ratings of movies. We convert the task data into binary classification data. The original user's rating for the item is a continuous value between 0 and 5. We mark the rating of 3.5 and above as 1 (positive), and mark scores below 3.5 as 0 (negative). The data are then divided into training and testing datasets based on user id. We randomly select users with about 80% of the data to enter the training set, and the rest of the users to enter the test set. The purpose is to test whether users would rate a given item above 3.5 (positive) based on historical behavior.

Besides, we also used K-fold cross-validation method based on time series to divide the dataset and did the corresponding comparative experiments. Since the sequence sorted by time cannot be disrupted, time-based K-fold cross-validation will inevitably result in a part of the validation set data not participating in training. After the K-fold cross-validation training is completed, we do another model training that includes the entire training set. Then, we choose the model with the smallest error in the validation set for each fold, and put the test set on the model for evaluation. Finally, we define model performance as the average error on the test set of the models selected in each fold of cross-validation.

Table 1 below lists the information on the datasets we used.

[1] https://www.yelp.com/dataset.

[2] https://grouplens.org/datasets/movielens/1m/.

[3] https://grouplens.org/datasets/movielens/10m/.

Table 1 Basic information of datasets

Dataset	User	Item	Ratings	Sparsity
Yelp	46,712	78,654	1,262,486	0.9997
MovieLens 1M	6040	3706	1,000,209	0.9553
MovieLens 10M	69,878	10,677	10,000,054	0.9866

Parameter settings

The hyperparameters used in this paper are shown in Table 2.

Table 2 Hyperparameters' configuration

Hidden_size	128	Dropout	0.1
n_layers	3	Epochs	100
n_fold	100	Batch_size	256
Weight_size	3×64	Learning rate	0.01

Baselines' models

To show the results achieved by the proposed model, we took the following baselines:

DIN [24]: the model designs a local activation unit in the deep interest network to adaptively learn the representation of the user's interest from the user's historical behavior toward an item.

DeepFM [25]: the model uses the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture.

Wide and deep [26]: the model utilizes the wide model to process manually labeled cross-product features, and the deep model to extract nonlinear relationships between features.

PNN [27]: the model introduces an additional product layer as a feature extractor.

HOM-LIN [8]: the model defines a new unexpected interest distance function and recommends a model of unexpected interest to users through a mixed utility function.

PURS [9]: the model multi-clusters modeling of user interest and personalized surprise in a latent space through a self-attention mechanism and choosing an appropriate surprise activation function.

Evaluation metrics

It is worth noting that there is currently no clear evaluation metric to measure the standard of unexpected recommendation. Different researchers have given different evaluation metrics in their papers. The baseline models we compare include not only unexpected interest recommendation models, but also other types of models. Therefore, we finally choose the traditional recommendation system evaluation metrics to evaluate our model after comprehensive consideration.

To verify the superiority of the model, we chose the following two metrics:

HR@K: The hit rate is calculated by collecting the first K pieces of data. This model uses HR@10, and the formula is as follows:

$$\mathrm{HR}@K = \frac{1}{N}\sum_{i=1}^{N}\mathrm{HITS}@K(i),$$

where N is the total number of users, HITS@K indicates whether the value accessed by the i-th user is in the top-K items, the hit is 1, otherwise it is 0.

precision@K: The precision represents the probability of correctly predicting a positive sample among the samples predicted as positive samples. Our model predicts the accuracy of the top ten items.

In addition to the above two metrics, we additionally use the AUC metric to observe the ranking loss of the model. However, since we choose to use binary data, the amount of information contained is limited, so the experimental results are for reference only. It is defined as follows:

AUC: Measure the accuracy of the recommendation order by ranking all items that predict click-through rate and comparing with click information. A variation of the user-weighted AUC is introduced in UIRS-GNN, which measures the goodness of the user's internal order by averaging the user's AUC. We employ this metric in our experiments. For simplicity, we will still refer to it as AUC. The definition is as follows:

$$\mathrm{AUC} = \frac{\sum_{i=1}^{n}\mathrm{impression}_i \times \mathrm{AUC}_i}{\sum_{i=1}^{n}\mathrm{impression}_i},$$

where n is the number of users, $\mathrm{impression}_i$ and AUC_i are the impressions and AUC of the i-th user.

Table 3 Comparison of experimental results

Model	Yelp			MovieLens 1m			MovieLens 10m		
	HR@10	precision@10	AUC	HR@10	precision@10	AUC	HR@10	precision@10	AUC
DIN*	0.5905	0.7362	0.6828	0.6909	0.6955	0.7283	0.6807	0.7140	0.7327
DeepFM*	0.5868	0.7285	0.6763	0.6921	0.6912	0.7280	0.6825	0.7132	0.7322
Wide&Deep*	0.5587	0.7333	0.6538	0.6918	0.6965	0.7281	0.6811	0.7135	0.7311
PNN*	0.5879	0.7361	0.6822	0.6843	0.6940	0.7251	0.6812	0.7138	0.7312
HOM-LIN*	0.5556	0.6715	0.5527	0.6620	0.6726	0.7043	0.6682	0.7004	0.7124
PURS*	0.5563	0.6726	0.6680	0.6927	0.6962	0.7284	0.6831	0.7171	0.7329
UIRS-GNN	**0.5960**	**0.7621**	**0.6870**	**0.7076**	**0.7052**	**0.7290**	**0.7157**	**0.8628**	**0.7331**

The best results for each dataset are shown in bold, where * is the result reproduced by the public code of the original paper in the experimental environment of this paper, and the others are quoted from the original paper

Table 4 Comparison of experimental results (K-fold cross-validation)

Model	Yelp			MovieLens 1m			MovieLens 10m		
	HR@10	precision@10	AUC	HR@10	precision@10	AUC	HR@10	precision@10	AUC
DIN*	0.4706	0.7020	0.6430	0.5806	0.6523	0.7214	0.5791	0.7081	0.7324
DeepFM*	0.4762	0.7208	0.6313	0.5811	0.6644	0.7296	0.5777	0.6989	0.7364
Wide&Deep*	0.4766	0.7156	0.6394	0.5803	0.6619	0.7319	0.5773	0.7040	0.7334
PNN*	0.4652	0.6898	0.6326	0.5808	0.6528	0.7219	0.5788	0.7078	0.7319
HOM-LIN*	0.4591	0.6774	0.5503	0.5220	0.5659	0.6316	0.5623	0.6974	0.7157
PURS*	0.4620	0.6909	0.6385	0.5813	0.6660	0.7357	0.5796	0.7186	0.7487
UIRS-GNN	**0.4847**	**0.7409**	**0.6493**	**0.5969**	**0.7036**	**0.7473**	**0.5944**	**0.7375**	**0.7588**

The best results for each dataset are shown in bold, where * is the result reproduced by the public code of the original paper in the experimental environment of this paper, and the others are quoted from the original paper.

Result

Comparative experimental results and analysis

We validate our model with several competitive baseline models on three real datasets, and the experimental results are shown in Tables 3, 4 and Fig. 8. In terms of the scoring standard HR@10, compared to the sub-optimal baseline, our model has improved by 0.93%, 2.15% and 4.77%, respectively, on the three real datasets Yelp, MovieLens 1m and MovieLens 10m. In terms of scoring standard precision@10, our model improves by 3.52%, 1.25% and 20.32%, respectively, compared to the sub-optimal baseline. In addition, in terms of AUC indicators, since our given dataset is binary data (labels are defined as 0 and 1) and the amount of information contained in it is limited, the model can only give the final ranking value through the mapping relationship between features and binary labels. It can be understood that the classification task is accepted during training, and the regression task is to be completed during testing. But it can still be seen that our model is slightly ahead of other models.

In the comparative experiments of K-fold cross-validation methods, our model also performs well. In terms of HR@10, compared to the sub-optimal baseline, our model improves by 1.70%, 2.68% and 2.55%, respectively, on datasets Yelp, MovieLens 1m and MovieLens 10m. In terms of precision@10, our model improves by 2.79%, 5.65% and 2.63% on the three datasets. Notably, our model also has 0.96%, 1.58% and 1.35% improvement in AUC. In summary, our model achieves a significant improvement over the baseline model in training with both ways of splitting the dataset.

Among all the baseline models, HOM-LIN performs unsatisfactory. This is because it focuses on the unexpectedness of the recommendation and ignores the characteristics of learning users. These reasons lead to the deviation of the recommended content from the topic due to the high unexpectedness.

DIN adaptively learns preferences and interests in user behavior sequences by designing a local activation unit. DeepFM uses deep learning methods to learn user features in an end-to-end manner, and then combines them with a factorization machine for recommendation. Wide&Deep learns

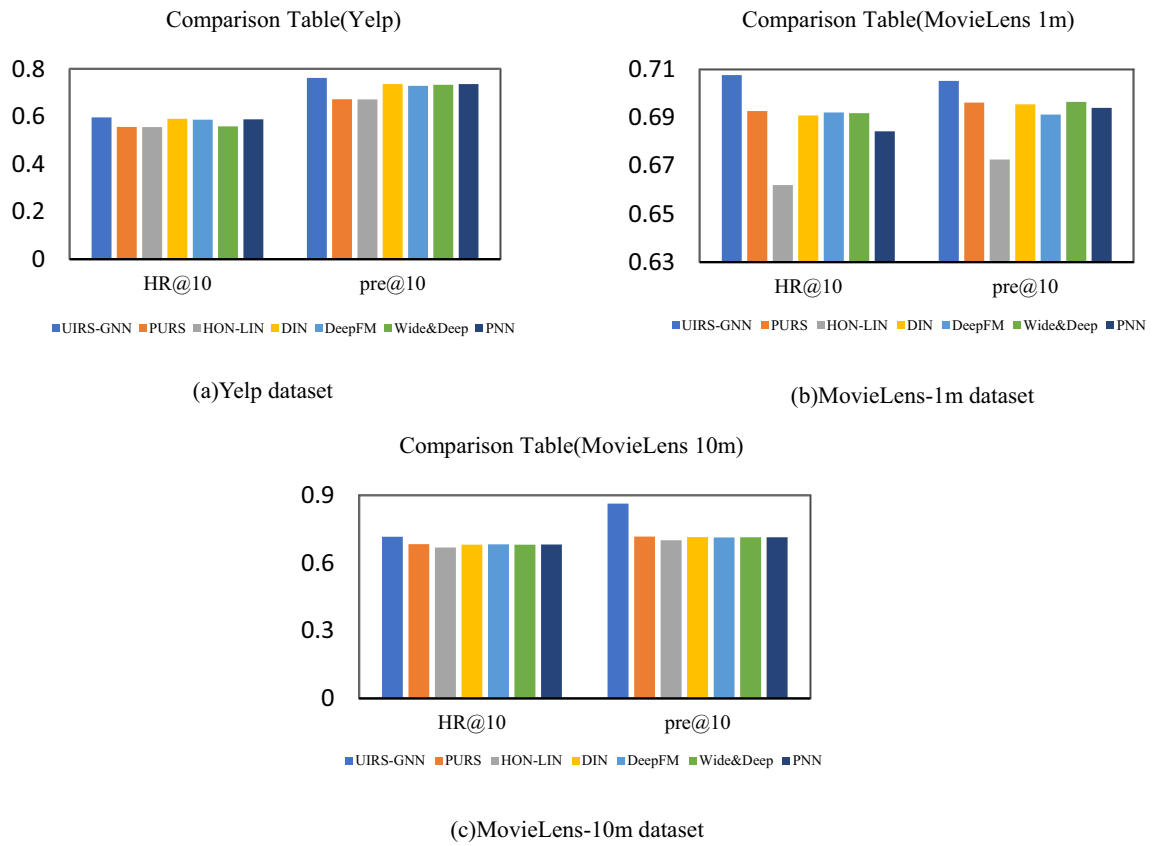


Fig. 8 Comparison of results of the model on three datasets

features through the Wide model, and uses the Deep model to learn the nonlinear relationship between features. It combines the benefits of recommender system memory and generalization. PNN mainly enriches node features by introducing product classification. Although they all made some breakthroughs in the relevance of recommended content, they ignored the over-specialization problem of recommendation.

PURS simultaneously pays attention to the problems of two types of models. It simultaneously learns user characteristics and the unexpectedness of recommended content in an end-to-end manner to improve user's satisfaction. However, since the PURS model only uses the most basic recommendation model in the process of learning user and item features, it leads to the problem of insufficient feature learning. Taking the Yelp dataset as an example, when the user interaction data is too sparse, PURS does not learn enough data. Its unexpected recommendation module cannot accurately discover the user's preferences and interests and has a negative impact.

To address this problem, our model introduces a graph neural network to deep learn the characteristics of users and items, and inputs these data into A-LSGRU and an unexpected interest recommendation module in an end-to-end manner. In addition, it can be seen that the richness of the user's historical behavior also affects the accuracy of the model to a certain extent. Compared using the Yelp dataset where user interaction behavior is sparse, our model achieves more significant improvements with the MovieLens dataset.

Ablation study

As we can see in the previous section, our model has a significant improvement over the baseline. This is because the graph neural network enriches the features of the target node, and A-LSGRU combines the learning of the user's long-term and short-term preferences. Moreover, our model also adds an expected interest recommendation model to pay attention to the unexpectedness of the recommendation.

In this section, we conduct ablation studies for four points:

- Version1: In this model, we no longer learn users' short-term preferences and interests, and only focus on users' long-term preferences and interests to obtain user characteristics.
- Version2: In this model, we do not use neighborhood aggregation operations to enrich user features, but directly use the original user and item data as the input content of the model.

Table 5 Ablation experiment

Model	Yelp			MovieLens 1m			MovieLens 10m		
	HR@10	precision@10	AUC	HR@10	precision@10	AUC	HR@10	precision@10	AUC
Version1	0.5930	0.7461	0.6856	0.7051	0.7019	0.7278	0.7149	0.8619	0.7321
Version2	0.5559	0.6719	0.6673	0.6998	0.6966	0.7283	0.6747	0.7049	0.7326
Version3	0.5907	0.7431	0.6837	0.7042	0.7006	0.7273	0.7110	0.8581	0.7320
Version4	0.5920	0.7405	0.6828	0.7043	0.7008	0.7274	0.7109	0.8578	0.7319
UIRS-GNN	**0.5960**	**0.7621**	**0.6870**	**0.7076**	**0.7052**	**0.7290**	**0.7157**	**0.8628**	**0.7331**

The best results for each dataset are shown in bold

Table 6 Influence of neighborhood layers

Layer	Yelp			MovieLens 1m			MovieLens 10m		
	HR@10	precision@10	AUC	HR@10	precision@10	AUC	HR@10	precision@10	AUC
Layer-1	0.5906	0.7605	0.6839	0.7065	0.7042	0.7283	0.7151	0.8612	0.7319
Layer-2	0.5917	0.7614	0.6854	0.7061	0.7036	0.7282	0.7152	0.8619	0.7320
Layer-3	**0.5960**	**0.7621**	**0.6870**	**0.7076**	**0.7052**	**0.7290**	**0.7157**	**0.8628**	**0.7331**
Layer-4	0.5924	0.7616	0.6840	0.7068	0.7048	0.7288	0.7155	0.8622	0.7328

The best results for each dataset are shown in bold

- Version3: In this model, we do not use the A-LSGRU to learn the user's long-term preference interest and short-term preference interest.
- Version4: In this model, we do not use the unexpected interest recommendation model, and only focus on the user's relevance recommendation.

The results are shown in Table 5.

As can be seen from Table 5, the model that remove graph neural networks performs worst. When we remove the graph neural network, the model exposes the problem of insufficient feature learning. The A-LSGRU also improve the accuracy of our model. As mentioned in "Evaluation metrics" above, unexpected interest recommendation model does not have a unified unexpected evaluation metric. We only used it as a part of our model and tested its effect on our model. It is obvious that it improves the score of our model. Our special operations for short-term interests also contribute to the model. Therefore, it can be seen that several components in our model contribute to our model, and removing any one will reduce the effect of the model to a certain extent.

The influence of neighborhood layers on the recommendation effect

In Table 6, we examine the influence of neighborhood layers on the model. We test the performance of the model with the [1–4] layer neighborhood, respectively. Its performance is as follows:

As can be seen from Table 6, the model works best when the number of neighborhood layers is 3, which is consistent with the results found in the paper [19]. This is because the number of neighborhood layers will lead to insufficient aggregated domain nodes, so that user features cannot be fully learned. On the other hand, too many domain layers can also lead to overfitting, which reduces the performance of the model. In contrast, we set the number of domain layers of the model to three layers.

Conclusions

In this paper, we propose an unexpected interest recommender system with graph neural network (UIRS-GNN). UIRS-GNN pays attention to the relevance and unexpectedness of user-recommended content, and intends to improve user satisfaction while recommending unexpected content to users. We use a graph convolutional neural network to learn the neighborhood features of users and items, and then use the A-LSGRU to learn the user's interest preferences. We map the learned content into the latent space. We model the unexpectedness metric as the weighted distance between the target item and the set of interests to discovering the unexpected interests of users. Finally, we combine the results of the A-LSGRU and the unexpected interest model to improve user

satisfaction. Our experimental results on the three datasets demonstrate the superiority of the UIRS-GNN model comparing with several competitive baseline models.

Author contributions The research results of this manuscript come from our joint collaborative research.

Funding This work was supported in part by the National Natural Science Foundation of China (no. 61972182).

Availability of data and materials The data we use comes from public data sets: https://www.yelp.com/dataset. https://grouplens.org/datasets/movielens/. Yelp dataset is a subset of businesses, reviews, and user data for use in personal, educational, and academic purposes. In this paper, Yelp dataset includes more than 1 million pieces of rating data for 78,654 items by 46,712 users. MovieLens public data sets are widely used in movie recommendation systems. In this paper, MovieLens-1m (ML-1m) and MovieLens-10m (ML-10m) with auxiliary information are selected as experimental data sets, with a score range of 1–5 points. ML-1m includes more than 1 million pieces of rating data for 3706 items by 6040 users. ML-10m includes more than 10 million pieces of rating data for 69,878 items by 10,677 users.

Code availability The code of our paper is temporarily not available.

Declarations

Conflict of interest To the best of our knowledge, the named authors have no conflict of interest, financial or otherwise.

Ethics approval Ethics approval was not required for this research.

Consent to participate No one participated in the study of the manuscript.

Consent for publication Written informed consent for publication was obtained from all participants.

References

1. Aggarwal CC (2016) Content-based recommender systems. In: Recommender Systems, Springer, Berlin, pp 139–166
2. Ekstrand MD, Riedl JT, Konstan JA (2011) Collaborative filtering recommender systems. Now Publishers Inc, Delft
3. Wang S, Cao L, Wang Y, Sheng QZ, Orgun MA, Lian D (2021) A survey on session-based recommender systems. ACM Comput Surv (CSUR) 54(7):1–38
4. Adamopoulos P, Tuzhilin A (2014) On over-specialization and concentration bias of recommendations: probabilistic neighborhood selection in collaborative filtering systems. In: Proceedings of the 8th ACM conference on recommender systems, pp 153–160
5. Kapoor K, Subbian K, Srivastava J, Schrater P (2015) Just in time recommendations: modeling the dynamics of boredom in activity streams. In: Proceedings of the eighth ACM international conference on web search and data mining, pp 233–242
6. Ziarani RJ, Ravanmehr R (2021) Serendipity in recommender systems: a systematic literature review. J Comput Sci Technol 36(2):375–396
7. Jiang S, Zhang L, Zhou N (2019) A survey of diversity in personalized recommendation systems. Softw Eng Appl 8(03):172–178
8. Adamopoulos P, Tuzhilin A (2014) On unexpectedness in recommender systems: Or how to better expect the unexpected. ACM Trans Intell Syst Technol (TIST) 5(4):1–32
9. Li P, Que M, Jiang Z, Hu Y, Tuzhilin A (2020) PURS: personalized unexpected recommender system for improving user satisfaction. In: Fourteenth ACM conference on recommender systems, pp 279–288
10. Vaswani A, Shazeer N, Parmar N, et al. (2017). Attention is all you need. In: NIPS, pp 6000–6010
11. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: NIPS, pp 1106–1114
12. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. In: ICLR
13. Hamilton WL, Ying R, Leskovec J (2017) Representation learning on graphs: methods and applications. IEEE Data Eng Bull 40(3):52–74
14. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
15. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: NIPS, pp 1024–1034
16. Li P, Tuzhilin A (2019) Latent modeling of unexpectedness for recommendations. In: RecSys (late-breaking results), pp 36–40
17. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J (2018) Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, pp 974–983
18. He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, pp 173–182
19. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M (2020) Lightgcn: simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 639–648
20. Linden G, Smith B, York J (2003) Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Comput 7(1):76–80
21. Lipton ZC, Berkowitz J, Elkan C (2015) A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019
22. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
23. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555
24. Zhou G, Zhu X, Song C et al (2018) Deep interest network for click-through rate prediction. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1059–1068
25. Guo H, Tang R, Ye Y, Li Z, He X (2017) DeepFM: a factorization-machine based neural network for CTR prediction. In: Proceedings

of the 26th international joint conference on artificial intelligence, pp 1725–1731
26. Cheng H T, Koc L, Harmsen J et al (2016) Wide and deep learning for recommender systems. In: Proceedings of the 1st workshop on deep learning for recommender systems, pp 7–10
27. Qu Y, Cai H, Ren K, Zhang W, Yu Y, Wen Y, Wang J (2016) Product-based neural networks for user response prediction. In: 2016 IEEE 16th international conference on data mining (ICDM), pp 1149–1154
28. Chen L., Yang, Y., Wang, N., Yang, K., Yuan, Q. (2019, May) How serendipity improves user satisfaction with recommendations? A large-scale user evaluation. In: The world wide web conference, pp 240–250
29. Zhang YC, Séaghdha DÓ, Quercia D, Jambor T (2012) Auralist: introducing serendipity into music recommendation. In: Proceedings of the fifth ACM international conference on web search and data mining, pp 13–22
30. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2016) Session-based recommendations with recurrent neural networks. In: Proceedings of the 4th international conference on learning representations
31. Davis PJ (1959) Leonhard Euler's integral: a historical profile of the gamma function: in memoriam: Milton abramowitz. Am Math Mon 66(10):849–869