Expert Systems With Applications 238 (2024) 122242

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Item Attribute-aware Graph Collaborative Filtering

Anchen Li, Xueyan Liu, Bo Yang *

College of Computer Science and Technology, Jilin University, Changchun, China
Key Laboratory of Symbolic Computation and Knowledge Engineer (Jilin University), Ministry of Education, Changchun 130012, China

ARTICLE INFO

Keywords:
Recommender systems
Collaborative filtering
Graph neural networks
Relation learning
Item attribute
Data mining

ABSTRACT

Collaborative filtering (CF) is a widely used technique in recommender systems. While many CF methods primarily focus on collaborative signals derived from user–item interactions, they often overlook other types of similarities (e.g., user similarity in loving actors, movie similarity in sharing genres, etc.). These similarities offer the fine-grained knowledge to understand users and items, which can complement the collaborative signal. In this paper, we introduce a model that harnesses these additional similarities for improved recommendations. Our approach involves constructing both a user and an item relational graph, based on multiple item attributes. These graphs reflect semantic similarities from various perspectives among users and items. Then, we develop a recommendation framework that employs a dual graph neural network, integrating these graphs into the recommendation process. We name our approach **I**tem **A**ttribute-aware **G**raph **C**ollaborative **F**iltering (IAGCF). Through evaluations on six real-world datasets, we found that IAGCF surpasses several state-of-the-art recommenders.

1. Introduction

The recommender system is one of the effective means to alleviate the issue of information overload in various web platforms, such as Netflix and Amazon. Predicting whether users will engage (e.g., purchasing or clicking) with items is a core of recommender systems (Ricci et al., 2011). Collaborative filtering (CF) is a dominant recommender solution, whose heuristic is that users with resemble past behaviors will share similar preferences for items (Hu et al., 2008; Li, Yang, Huo, Chen et al., 2022).

As a traditional CF model, matrix factorization (MF) first deploys latent vectors to represent users and items, and then the inner product of embeddings is utilized to learn interactions (Koren et al., 2009). Recently, since deep learning has yielded success (Wu et al., 2011), researchers use multiple neural architectures to devise recommenders (Zhang et al., 2019). For example, NeuCF is a representative deep CF model which uses a multi-layer perceptron for CF tasks (He et al., 2017). As graph representation learning developed, some recent studies adopt graph convolutions to model high-order user–item connections (He et al., 2020; Wang, He, Wang et al., 2019).

Despite their prevalence, we believe that the above approaches often operate at a macro level as they consider only collaborative signals (i.e., relations evidenced by the user's history). However, real-world scenarios present multiple semantic similarities exist among users and items, offering supplementary insights to these collaborative signals. Fig. 1 shows a movie scenario, and we would like to recommend movies to user b. When only collaborative signals are used, movie 'Braveheart' may be a choice because user b and user a have resemble historical interactions. If we consider semantic similarities, movie 'Titanic' that user c likes can be recommended to user b as user b and user c may be similar in loving movie starring 'Leonardo DiCaprio'. In addition, since user b's favorite movie 'Inception' is science fiction, user b may also like another science fiction movie 'Avatar'. By integrating these semantic similarities, recommendation content becomes not only diverse but also reasonable. Thus, harnessing these similarities provides a richer understanding of user and item characteristics, enhancing the efficacy of recommender systems.

In this work, we build a recommender that relies on the following insight: attribute-aware similarities provide complementary functionality and additional insights for predicting a user's preference for an item. Here, attribute-aware user similarities refer to people having similar preferences on one or more item attributes, and attribute-aware item similarities mean items sharing one or more item attributes, as shown in Fig. 1. However, harnessing these insights is not straightforward due to several technical challenges. Firstly, users/items exhibit attribute-aware similarities from different aspects. The challenge lies in effectively capturing these nuances. Secondly, to support a detailed analysis, how to model similarity strengths? Lastly, during generating recommendations, the influence of similarities changes with users/items. How to design a model that accounts for these influences?

* Corresponding author at: College of Computer Science and Technology, Jilin University, Changchun, China.
E-mail addresses: liac20@mails.jlu.edu.cn (A. Li), xueyanliu@jlu.edu.cn (X. Liu), ybo@jlu.edu.cn (B. Yang).

https://doi.org/10.1016/j.eswa.2023.122242
Received 10 June 2023; Received in revised form 18 October 2023; Accepted 18 October 2023
Available online 25 October 2023
0957-4174/© 2023 Elsevier Ltd. All rights reserved.

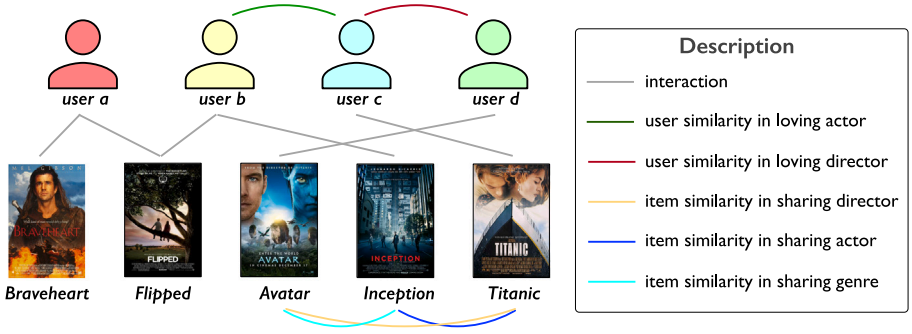


Fig. 1. An example for illustrating the similarity relation of users and items.

To solve these challenges, we present an approach IAGCF which stands for **I**tem **A**ttribute-aware **G**raph **C**ollaborative **F**iltering. IAGCF consists of two main components. First, IAGCF utilizes multiple attributes of items to construct user and item similarity matrices from different aspects. Based on these matrices, several measurement functions are adopted to define the similarity strength. After that, we derive a user relational graph and an item relational graph. These graphs reflect top-T attribute-aware similar neighbors for users and items, respectively, according to the definition of similarity strength. In the second component, IAGCF carries a dual graph neural network which provides an elegant way to integrate the constructed graphs for recommendation tasks. Specially, we devise an attention mechanism in the neighbor aggregation, which captures semantic influences of users/items for different aspect similarities. Empirically, we apply our IAGCF to six real-world scenarios. The results show that IAGCF outperforms the state-of-the-art approaches in click-through rate prediction and top-K recommendation.

In summary, our main contributions are listed as follows:

- We introduce a strategy for identifying similar users and items by item attributes. This involves constructing user and item relational graphs, which captures attribute-based similarities among users and items.
- We design a dual graph convolution network, integrating relational graphs to enhance recommendations. Furthermore, an attention mechanism is employed during neighbor aggregation to capture neighbor influences.
- Extensive experiments on six real-world datasets demonstrate that our methodology outperforms strong approaches in both click-through rate prediction and top-K recommendation tasks.

The remainder of this paper is structured as follows: Section 2 details the introduction of our recommender IAGCF. Section 3 discusses the time complexity analysis and IAGCF's relations with several existing models. Section 4 provides the experimental results accompanied by the analysis. Relevant literature pertaining to IAGCF is discussed in Section 5. Finally, Section 6 concludes the paper and outlines avenues for future research.

2. Methodology

In this section, we first introduce notations and formulate problems. We then present our method IAGCF, which contains two parts. The first part is the graph construction which builds up user and item relational graphs. The second part is a recommendation framework that utilizes a dual graph neural network to integrate the constructed relational graphs for recommendation.

2.1. Preliminary

This subsection introduces notations and formulate problems. Key notations are summarized in Table 1.

2.1.1. Notations

Let $\mathcal{U} = \{u_1, \dots, u_M\}$ and $\mathcal{V} = \{v_1, \dots, v_N\}$ represent user and item sets. We define $\mathbf{Y} \in \mathbb{R}^{M \times N}$ as the interaction matrix. The element $y_{ij} \in \{0, 1\}$ in $\mathbf{Y}$ represents whether u_i has clicked or purchased v_j. Moreover, there is an attribute set which is associated with items, donated as $\mathcal{A} = \{a_1, \dots, a_K\}$. The element $a_i \in \mathcal{A}$ is a tuple (t_i, E_i), where t_i and E_i indicate type and values of attribute a_i. Since a_i could be a single-value or a multi-value attribute, we define E_i as $\{e_{i1}, \dots, e_{iL_i}\}$, whose element is an attribute value and L_i is the number of values in E_i. Taking movies for example, *comedy* (attribute value) and *adventure* (attribute value) are in the *genres* (attribute type)'s attribute value set.

2.1.2. Problem formulation

Given the information $(\mathcal{U}, \mathcal{V}, \mathbf{Y}, \mathcal{A})$, IAGCF first outputs a user relational graph $\mathbf{G}_u$ and an item relational graph $\mathbf{G}_v$, which are defined as heterogeneous graphs composed of triples, as follows: $\mathbf{G}_u = \{(u_a, r_u, u_b) | u_a, u_b \in \mathcal{U}, r_u \in \mathcal{R}_u\}$ and $\mathbf{G}_v = \{(v_a, r_v, v_b) | v_a, v_b \in \mathcal{V}, r_v \in \mathcal{R}_v\}$. Here, $\mathcal{R}_u$ and $\mathcal{R}_v$ ($|\mathcal{R}_u| = |\mathcal{R}_v| = K$) are relation sets, which contain user and item similarity relations. After the graph construction, given the information ($\mathbf{Y}$, $\mathbf{G}_u$, $\mathbf{G}_v$), IAGCF designs a dual graph neural network which aims to learn a prediction function $\hat{y}_{uv} = \mathcal{F}(u, v | \Theta, \mathbf{Y}, \mathbf{G}_u, \mathbf{G}_v)$, where $\hat{y}_{uv}$ is the probability that whether user u will engage item v which he has never interacted with, and Θ is the parameters of $\mathcal{F}$.

2.2. Graph construction

The crux of the graph construction is to solve the following challenges. Take the user side for example: (i) Note that each user has attribute-aware similarities from different perspectives. Fig. 2 gives an example in the movie domain, user a and user b are similar in loving the actor attribute, while user a and user c are similar in loving the genre attribute. How to capture different aspects of attribute-aware similarities for users and items is the first challenge. (ii) To support a fine-grained analysis, we should allow for different strengths in similarity. For instance, in Fig. 2, user a and user b are similar to user c in the genre attribute with different similarity strengths. How to model the different similarity strengths for users and items is another challenge.

To solve these challenges, we propose a graph construction strategy, which contains three major steps: (i) construct item attribute matrices and user preference matrices; (ii) construct item and user similarity strength matrices; and (iii) construct user and item relational graphs. Next we detail each step.

2.2.1. Step 1: Construction of attribute and preference matrices

For a given item v_c, we use a feature vector $p_{ci} \in \mathbb{R}^{1 \times L_i}$ to characterize its attribute a_i. Specifically, if v_c associates with the rth attribute value e_{ir} of attribute a_i, then the rth value in p_{ci} is 1, otherwise is 0. In this way, for attribute a_i, we form an item attribute matrix $P_i = [p_{1i}; p_{2i}; \dots; p_{Ni}]$ whose element is an item's attribute vector. For a

Table 1
Key notations in this work.

Symbols	Definitions	Symbols	Definitions
$\mathcal{U}$	The user set	$\mathcal{V}$	The item set
M	The number of users	N	The number of items
$\mathbf{Y}$	The interaction matrix	$\mathcal{A}$	The attribute set
K	The number of attributes	t_i	The attribute type for attribute a_i
E_i	The attribute value set for attribute a_i	L_i	The number of attribute values for a_i
P_i	The item attribute matrix for attribute a_i	Q_i	The user preference matrix for attribute a_i
S_i^v	The item similarity strength matrix for a_i	S_i^u	The user similarity strength matrix for a_i
$\mathbf{G}_v$	The item relational graph	$\mathbf{G}_u$	The user relational graph
$\mathcal{N}_i^v$	The similar neighbor set of item v_i	$\mathcal{N}_a^u$	The similar neighbor set of user u_a
$\mathcal{R}_v$	The item similarity relation set	$\mathcal{R}_u$	The user similarity relation set
$\mathbf{E}_v$	The item embedding matrix	$\mathbf{E}_u$	The user embedding matrix
$\mathbf{R}_v$	The item relation embedding matrix	$\mathbf{R}_u$	The user relation embedding matrix
H	The number of aggregation layers	d	The embedding size
Θ	The trainable parameters	$\hat{y}_{uv}$	The predicted probability from user u to item v

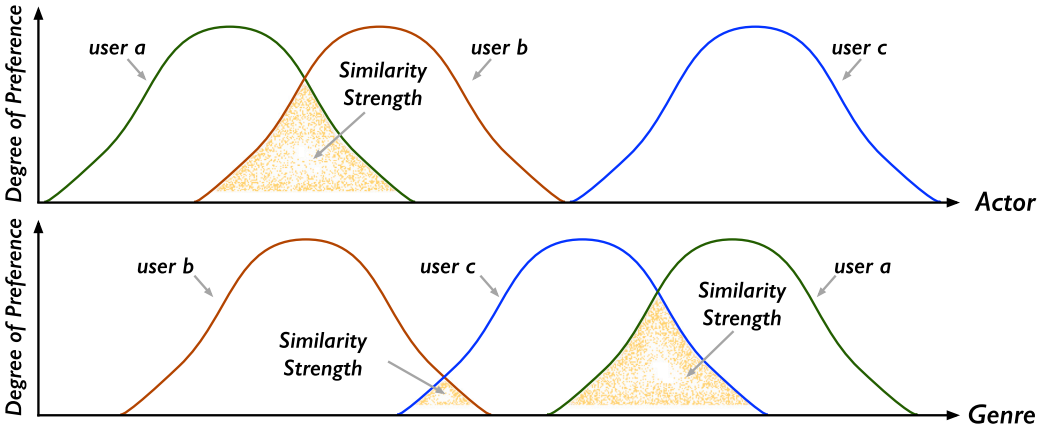


Fig. 2. Examples of the challenges in IAGCF's graph construction strategy.

given user u_d, we construct a vector $q_{di} \in \mathbb{R}^{1\times L_i}$ to characterize the u_d's preference for item attribute a_i. Specifically, if user u_d engaged with the item v_c in the historical records, then the attribute vector p_{ci} of item v_c is added to q_{di}. In this way, if user u_d is associated with e_{ir} of attribute a_i for h times, then the rth value is h in q_{di}. Similar to items, we obtain a user preference matrix $Q_i = [q_{1i}; q_{2i}; \ldots; q_{Mi}]$ on item attribute a_i, and the element in Q_i is regarded as a user preference vector for attribute a_i. At last, an item attribute matrix set $P = \{P_1, P_2, \ldots, P_K\}$ and a user preference matrix set $Q = \{Q_1, Q_2, \ldots, Q_K\}$ are obtained. Fig. 3(b) and (c) are examples of an item attribute matrix and a user preference matrix for the actor attribute based on the user–item interaction of Fig. 3(a).

2.2.2. Step 2: Construction of strength matrices

In this step, we construct user and item similarity strength matrices by leveraging constructed item attribute matrices and user preference matrices. We first define the similarity strength.

Given an item attribute matrix P_i, the item similarity strength for the ith attribute between item v_a and item v_b is defined as: $\mathcal{S}(p_{ai}, p_{bi})$. Similarly, given a user preference matrix Q_i, the user similarity strength for the ith attribute between user u_a and user u_b is defined as: $\mathcal{S}(q_{ai}, q_{bi})$. Here, function $\mathcal{S}$ is utilized to calculate the similarity between two vectors. We would like the similarity strength to satisfy two properties, i.e., *symmetric*: $\mathcal{S}(a,b) = \mathcal{S}(b,a)$ and *self-maximum*: $\mathcal{S}(a,b) \le \mathcal{S}(a,a) = \mathcal{S}(b,b)$. Here, we consider three functions: (a) heat kernel $\mathcal{S}_h(\cdot,\cdot)$, (b) cosine similarity $\mathcal{S}_c(\cdot,\cdot)$, and (c) Jensen–Shannon divergence $\mathcal{S}_j(\cdot,\cdot)$, as follows:

$$\begin{aligned} &\text{(a)}: \ \mathcal{S}_h(a,b) = e^{-\frac{\|a-b\|^2}{t}}, \quad \text{(b)}: \ \mathcal{S}_c(a,b) = \frac{a\cdot b}{|a\parallel b|}, \\ &\text{(c)}: \ \mathcal{S}_j(a,b) = 1 - \frac{1}{2}\left(\mathcal{K}(\bar{a}\parallel \frac{\bar{a}+\bar{b}}{2}) + \mathcal{K}(\bar{b}\parallel \frac{\bar{a}+\bar{b}}{2})\right), \end{aligned} \tag{1}$$

in which t is the hyper-parameter for the heat kernel, $\mathcal{K}(P \parallel Q) = \sum P(x)\log\frac{P(x)}{Q(x)}$ is Kullback–Leibler divergence, and $\bar{a}$, $\bar{b}$ are normalization vectors for a, b. We can see that two items/users having a higher strength value shows that they are more similar. In this paper, we uniformly use the cosine similarity as the similarity function.

Based on the definition of the similarity strength, for the attribute a_i, we obtain an item similarity strength matrix $S_i^v \in \mathbb{R}^{N\times N}$ and a user similarity strength matrix $S_i^u \in \mathbb{R}^{M\times M}$. Each element $S_i^v(jk) \in S_i^v$ (or $S_i^u(jk) \in S_i^u$) is the similarity strength from item v_j to item v_k (or from user u_j to user u_k). Fig. 3(d) and (e) are examples of item and user similarity strength matrices based on Fig. 3(b) and (c), where colors represent different strengths.

2.2.3. Step 3: Construction of user and item relational graphs

Given the strength matrices of users and items, similar items and users can be found. Specifically, we choose top-T strength users to be the attribute-aware similar users for each user and link them to construct the user relational graph. The construction of the item relational graph is similar to the user. The algorithm for the graph construction strategy is detailed in Algorithm 1. Note that the construction of user and item graphs is constrained to using the interaction records in the training split. LINK function is used to link nodes (similar users or items) in the graphs. Specifically, if there are multiple nodes that have the same strength with the target node, we choose one with ties broken arbitrarily. Fig. 3(f) is an example of item and user relational graphs when T = 1.

In this way, we construct two heterogeneous graphs as user and item relational graphs, in which nodes represent users/items and edges represent different types of attribute-aware similarities between users/items.

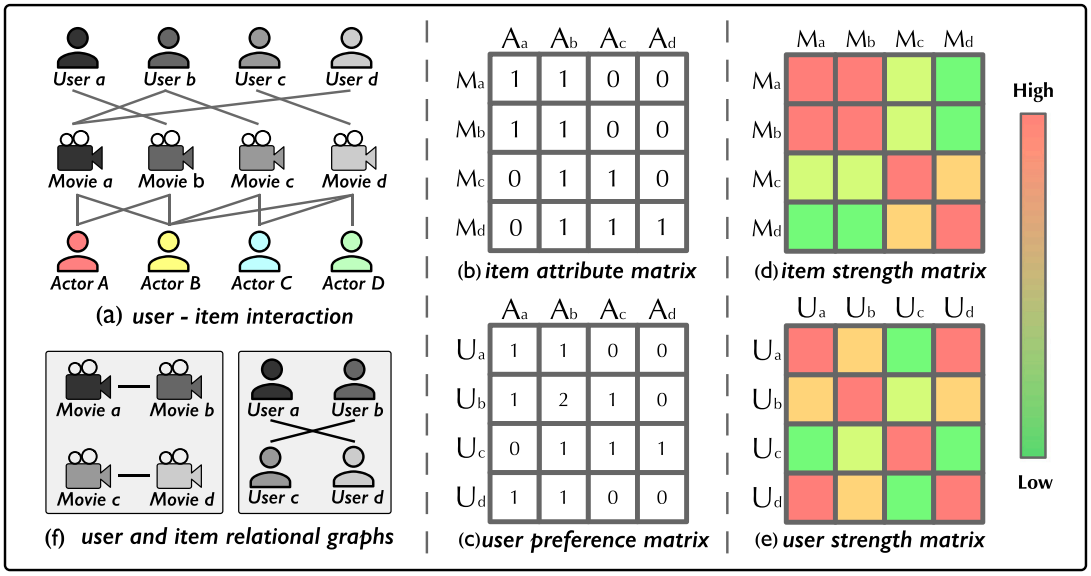


Fig. 3. Examples of the (a) user–item interaction, (b) item attribute matrix, (c) user preference matrix, (d) item similarity strength matrix, (e) user similarity strength matrix and (f) item and user relational graphs.

Algorithm 1: Construction of the item and user relational graphs

Input : User set $\mathcal{U}$; item set $\mathcal{V}$; item attribute set $\mathcal{A}$; interaction matrix $\mathbf{Y}$; sampling size T.
Output: Item relational graph $\mathbf{G}_v$; user relational graph $\mathbf{G}_u$.

$\mathbf{G}_v = \mathbf{G}_u = \emptyset$;
for *each attribute* $a_i \in \mathcal{A}$ **do**
  $G_v^i = G_u^i = \emptyset$;
  calculate item attribute matrix P_i based on $\mathcal{A}$, $\mathcal{V}$;
  calculate item similarity strength matrix S_i^v based on P_i;
  for *each item* $v_j \in V$ **do**
    $G_v^i = G_v^i \cup \text{LINK}(v_j, S_i^v, \text{T}, a_i)$;
  end
  $\mathbf{G}_v = \mathbf{G}_v \cup G_v^i$;
  calculate user preference matrix Q_i based on P_i, $\mathbf{Y}$, $\mathcal{U}$;
  calculate user similarity strength matrix S_i^u based on Q_i;
  for *each user* $u_k \in U$ **do**
    $G_u^i += G_u^i \cup \text{LINK}(u_k, S_i^u, \text{T}, a_i)$;
  end
  $\mathbf{G}_u = \mathbf{G}_u \cup G_u^i$;
end
Return $\mathbf{G}_v$, $\mathbf{G}_u$.
// Link similar users or items
Function LINK(w_h, S_i, T, a_i)
$\mathcal{T} = \emptyset$; // Initialize the triple set
for *number of iterations in T* **do**
  $o = \text{Pick_One}(\{\arg\max(S_i(h*))\})$;
  // Link w_h and w_o with relation r_i.
  $\mathcal{T} = \mathcal{T} \cup \{(w_h, r_i, w_o)\}$;
  $S_i(ho) = -1$;
end
Return $\mathcal{T}$.

2.3. Recommendation framework

This subsection first presents our recommendation framework of IAGCF, and then introduces the framework's optimization process. The framework takes the interaction history and two relational graphs as inputs and outputs the prediction preference from users to items. Specifically, the framework consists of three parts, as shown in Fig. 4. First, we leverage an embedding layer to initialize the user nodes, item nodes, and similarity relations from relational graphs. Then, the aggregation layers are introduced to refine the embeddings on user and item relational graphs, respectively. An attention mechanism is introduced in the aggregation layers to capture the semantic influence of users/items for different similarity relations. Finally, the prediction layer outputs the preference scores from users to items based on the refined embeddings. In what follows, details of each part are elaborated.

2.3.1. Embedding layer

Such layer encodes nodes and relations from relational graphs with embeddings. For user and item nodes, we utilize embedding matrices $\mathbf{E}_u \in \mathbb{R}^{M\times d}$ and $\mathbf{E}_v \in \mathbb{R}^{N\times d}$ to describe them, respectively, as follows:

$$\mathbf{E} = [\mathbf{E}_u, \mathbf{E}_v] = [\underbrace{\mathbf{e}_{u_1}, \mathbf{e}_{u_2}, \cdots \mathbf{e}_{u_M}}_{\text{User Embeddings}}, \underbrace{\mathbf{e}_{v_1}, \mathbf{e}_{v_2}, \cdots \mathbf{e}_{v_N}}_{\text{Item Embeddings}}], \tag{2}$$

where $\mathbf{e}_{u_i} \in \mathbb{R}^d$ and $\mathbf{e}_{v_j} \in \mathbb{R}^d$ are embeddings for user u_i and item v_j, respectively, and d is the embedding size.

In addition, we introduce embedding matrices $\mathbf{R}_u \in \mathbb{R}^{K\times d}$ and $\mathbf{R}_v \in \mathbb{R}^{K\times d}$ for initializing the features of user and item similarity relations in relational graphs, respectively, as follows:

$$\mathbf{R} = [\mathbf{R}_u, \mathbf{R}_v] = [\underbrace{\mathbf{r}_u^1, \mathbf{r}_u^2, \cdots \mathbf{r}_u^K}_{\text{User Relations}}, \underbrace{\mathbf{r}_v^1, \mathbf{r}_v^2, \cdots \mathbf{r}_v^K}_{\text{Item Relations}}], \tag{3}$$

where $\mathbf{r}_u^i \in \mathbb{R}^d$ and $\mathbf{r}_v^j \in \mathbb{R}^d$ are the ith user similarity relation and jth item similarity relation in user and item relational graphs, respectively.

2.3.2. Aggregation layer

To consider user and item neighbors in relational graphs, we propose dual aggregation operations. Since the aggregation process of users and items are the same, we describe the user aggregation, as shown in Fig. 5. For a given user u_a and her neighbors $\mathcal{N}_a^u$ from $\mathbf{G}_u$, the aggregation is a two-stage process, as follows:

$$\mathbf{e}_{u_a}^{agg} = \text{Agg}_2(u_a, \text{Agg}_1(\mathcal{N}_a^u)), \tag{4}$$

where process Agg_1 is utilized to transform and aggregate the neighbor features, and process Agg_2 is utilized to combine the neighbor features with the central node representations. A direct idea is using the mean pooling and summation operation for Agg_1 and Agg_2, respectively, as follows:

$$\mathbf{e}_{u_a}^{agg} = \mathbf{e}_{u_a} + \frac{1}{|\mathcal{N}_a^u|}\sum_{u_b\in\mathcal{N}_a^u}\mathbf{e}_{u_b}, \tag{5}$$

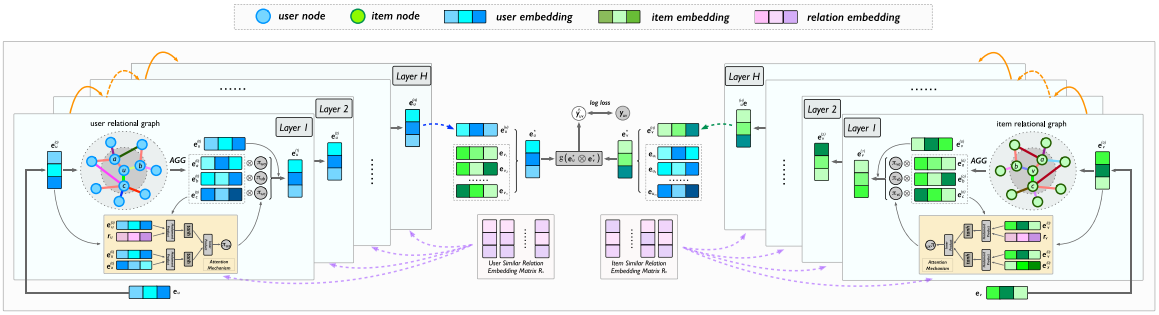


Fig. 4. Overall structure of IAGCF's recommendation framework. Blue nodes and green nodes represent users and items, respectively. Different colored edges in relational graphs represent different similarity relations between users/items. The light yellow block in the aggregation layers is the attention network. The framework takes relational graphs and the interaction history as inputs, then outputs the final prediction preference from users to items.

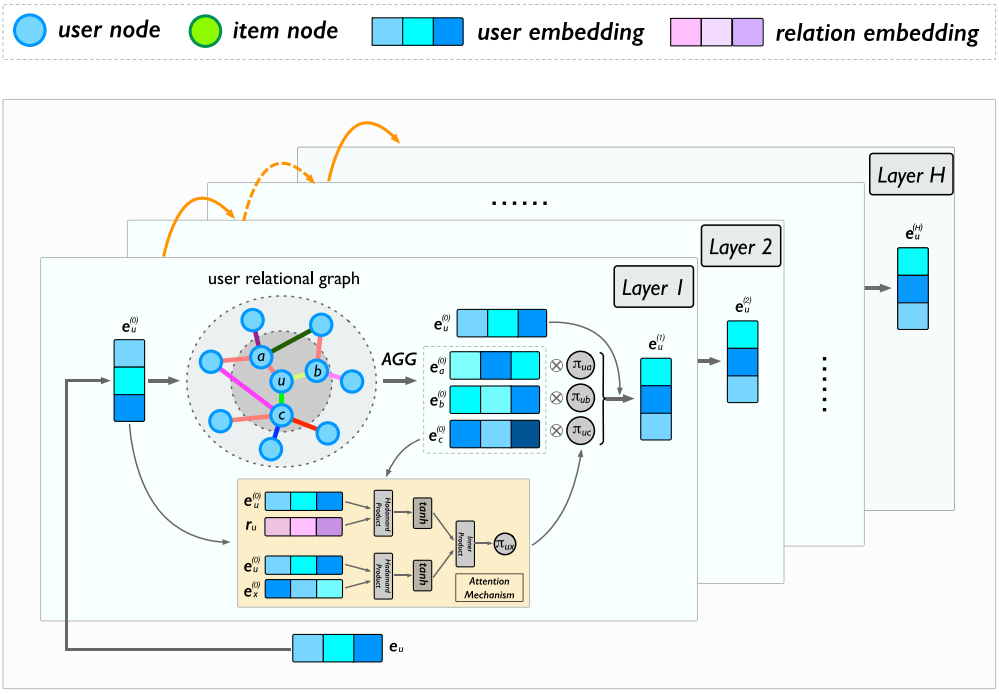


Fig. 5. Description of the user aggregation process.

However, such naive aggregation for neighbors neglects the relations information in graphs. In addition, as mentioned earlier, the importance and meaning of different relations vary for different users. Therefore, to learn the weight of neighbors, we adopt an attention mechanism in the aggregation process, which considers both neighbor and relation information. Specifically, for a triplet $(u_a, r_u^{ab}, u_b) \in \mathbf{G}_u$, the attention operation is defined as:

$$\pi_u^{ab} = \frac{\exp(\tanh(\mathbf{e}_{u_a} \odot \mathbf{e}_{u_b})^\top \tanh(\mathbf{e}_{u_a} \odot \mathbf{r}_u^{ab}))}{\sum_{u_{b'} \in \mathcal{N}_a^u} \exp(\tanh(\mathbf{e}_{u_a} \odot \mathbf{e}_{u_{b'}})^\top \tanh(\mathbf{e}_{u_a} \odot \mathbf{r}_u^{ab'}))}, \tag{6}$$

where $\odot$ is the Hadamard product and $\mathbf{r}_u^{ab} \in \mathbf{R}_u$ is the similarity relation embedding for relation r_u^{ab}.

In this way, the aggregation process is formulated by the weighted sum, as follows:

$$\mathbf{e}_{u_a}^{agg} = \mathbf{e}_{u_a} + \sum_{u_b \in \mathcal{N}_a^u} \pi_u^{ab} \mathbf{e}_{u_b}. \tag{7}$$

We can further repeat such aggregation multiple times to exploit the high-order connections. This process is similar to items. Specifically, in the hth aggregation, user u_a's and item v_c's updated embeddings $\mathbf{e}_{u_a}^{(h)}$ and $\mathbf{e}_{v_i}^{(h)}$ is modeled as:

$$\mathbf{e}_{u_a}^{(h)} = \mathbf{e}_{u_a}^{(h-1)} + \sum_{u_b \in \mathcal{N}_a^u} \pi_u^{ab} \mathbf{e}_{u_b}^{(h-1)}, \tag{8}$$

$$\mathbf{e}_{v_i}^{(h)} = \mathbf{e}_{v_i}^{(h-1)} + \sum_{v_j \in \mathcal{N}_i^v} \pi_v^{ij} \mathbf{e}_{v_j}^{(h-1)}. \tag{9}$$

Since some recent studies show that the linear transformation may degrade the performance of graph-based recommenders (Chen et al., 2020; He et al., 2020), we discard this operation in the process of updating representations. Finally, we obtain user embedding $\mathbf{e}_{u_a}^{(H)}$ and item embedding $\mathbf{e}_{v_i}^{(H)}$ after stacking H aggregation layers.

2.3.3. Prediction layer

To model final embeddings, we consider not only similarity relations, but also the historical interaction. Specifically, for a candidate user–item pair (u_a, v_i), user final embedding $\mathbf{e}_{u_a}^*$ and item final embedding $\mathbf{e}_{v_i}^*$ are defined as:

$$\mathbf{e}_{u_a}^* = \mathbf{e}_{u_a}^{(H)} + \frac{1}{|C_a^u|} \sum_{v_c \in C_a^u} \mathbf{e}_{v_c}, \tag{10}$$

$$\mathbf{e}_{v_i}^* = \mathbf{e}_{v_i}^{(H)} + \frac{1}{|C_i^v|} \sum_{u_c \in C_i^v} \mathbf{e}_{u_c}, \tag{11}$$

where C_a^u is the item set that user u_a engaged and C_i^v is the user set that item v_i is engaged by. In this way, historical behaviors and similarity relations complement each other, which enhances the expressive for the final embeddings. After that, we leverage a prediction function $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ to model the interaction, as follows:

$$\hat{y}_{ai} = g(\mathbf{e}^*_{u_a}, \mathbf{e}^*_{v_i}). \tag{12}$$

Here, g is defined as the inner product, and other operations (e.g., multi-layer perceptron) can also be applied.

2.3.4. Learning algorithm

To estimate network parameters Θ, we intend to maximize the following posterior probability:

$$\Theta = \arg\max_{\Theta} p(\Theta|\mathbf{G}_u, \mathbf{G}_v, \mathbf{Y}), \tag{13}$$

where $\mathbf{G}_u$ and $\mathbf{G}_v$ are the user and item relational graphs, and $\mathbf{Y}$ is the interaction matrix. Θ is the parameters in the framework, which consists of user and item embeddings $\mathbf{E} = [\mathbf{E}_u, \mathbf{E}_v]$ and relation embeddings $\mathbf{R} = [\mathbf{R}_u, \mathbf{R}_v]$.

Based on the Bayes theorem, Eq. (13) is equivalent to maximizing the following equation:

$$\begin{aligned} p(\Theta|\mathbf{G}_u, \mathbf{G}_v, \mathbf{Y}) &= \frac{p(\Theta, \mathbf{G}_u, \mathbf{G}_v, \mathbf{Y})}{p(\mathbf{G}_u, \mathbf{G}_v, \mathbf{Y})} \propto p(\Theta, \mathbf{G}_u, \mathbf{G}_v, \mathbf{Y}) \\ &= p(\Theta) \cdot p(\mathbf{G}_u, \mathbf{G}_v|\Theta) \cdot p(\mathbf{Y}|\Theta, \mathbf{G}_u, \mathbf{G}_v). \end{aligned} \tag{14}$$

The first term $p(\Theta)$ is the priori probability of parameters Θ, which includes the user and item embedding $\mathbf{E}$ and relation embedding $\mathbf{R}$. Following the way in Glorot and Bengio (2010), the parameters are initialized by the Xavier initializer.

The second term $p(\mathbf{G}_u, \mathbf{G}_v|\Theta)$ is the likelihood function of graphs $\mathbf{G}_u$ and $\mathbf{G}_v$ given corresponding parameters, as:

$$\begin{aligned} p(\mathbf{G}_u, \mathbf{G}_v|\Theta) =& p(\mathbf{G}_u, \mathbf{G}_v|\mathbf{E}_u, \mathbf{E}_v, \mathbf{R}_u, \mathbf{R}_v) \\ =& p(\mathbf{G}_u|\mathbf{E}_u, \mathbf{R}_u) \cdot p(\mathbf{G}_v|\mathbf{E}_v, \mathbf{R}_v). \end{aligned} \tag{15}$$

To estimate the likelihood, we develop a special task for relational graphs $\mathbf{G}_u$ and $\mathbf{G}_v$. Since relations is symmetrical, we utilize the idea of DistMult model (Yang et al., 2015) for the likelihood, as follows:

$$\begin{aligned} p(\mathbf{G}_u|\mathbf{E}_u, \mathbf{R}_u) &= \prod_{(u_a, r_u^{ab}, u_b) \in \mathcal{U} \times R_u \times \mathcal{U}} p\left((u_a, r_u^{ab}, u_b)|\mathbf{E}_u, \mathbf{R}_u\right) \\ &= \prod_{(u_a, r_u^{ab}, u_b) \in \mathcal{U} \times \mathcal{R}_u \times \mathcal{U}} \mathcal{N}\left(I_{u_a, r_u^{ab}, u_b} - \mathbf{u}_a^\top \tilde{\mathbf{r}}_u^{ab} \mathbf{u}_b, \lambda_u^{-1}\right), \end{aligned} \tag{16}$$

$$\begin{aligned} p(\mathbf{G}_v|\mathbf{E}_v, \mathbf{R}_v) &= \prod_{(v_a, r_v^{ab}, v_b) \in \mathcal{V} \times R_v \times \mathcal{V}} p\left((v_a, r_v^{ab}, v_b)|\mathbf{E}_v, \mathbf{R}_v\right) \\ &= \prod_{(v_a, r_v^{ab}, v_b) \in \mathcal{V} \times \mathcal{R}_v \times \mathcal{V}} \mathcal{N}\left(I_{v_a, r_v^{ab}, v_b} - \mathbf{v}_a^\top \tilde{\mathbf{r}}_v^{ab} \mathbf{v}_b, \lambda_v^{-1}\right), \end{aligned} \tag{17}$$

where $\tilde{\mathbf{r}}_u^{ab} = \mathrm{diag}(\mathbf{r}_u^{ab})$ and $\tilde{\mathbf{r}}_v^{ab} = \mathrm{diag}(\mathbf{r}_v^{ab})$ is the diagonal matrices for $\mathbf{r}_u^{ab}$ and $\mathbf{r}_v^{ab}$; $\mathcal{N}$ represents the Gaussian distribution; I_{u_a, r_u^{ab}, u_b} and I_{v_a, r_v^{ab}, v_b} are indicators, e.g., $I_{u_a, r_u^{ab}, u_b} = 1$ if $(u_a, r_u^{ab}, u_b) \in \mathbf{G}_u$ and $I_{u_a, r_u^{ab}, u_b} = 0$ otherwise.

The last term $p(\mathbf{Y}|\Theta, \mathbf{G}_u, \mathbf{G}_v)$ in Eq. (14) is the likelihood function of the observed feedback $\mathbf{Y}$ given corresponding parameters, user relational graph $\mathbf{G}_u$ and item relational graph $\mathbf{G}_v$. For the user–item interaction $y_{uv} \in \mathbf{Y}$, we use the product of Bernoulli distributions to define the likelihood, as:

$$p(\mathbf{Y}|\Theta, \mathbf{G}_u, \mathbf{G}_v) = \prod_{y_{uv} \in \mathbf{Y}} \sigma(\hat{y}_{uv})^{y_{uv}} \cdot (1 - \sigma(\hat{y}_{uv}))^{1 - y_{uv}}, \tag{18}$$

where $\hat{y}_{uv}$ is prediction score for user–item pair (u, v) according to Eq. (12).

Finally, we obtain the objective function by using the negative logarithm for Eq. (14), as follows:

$$\begin{aligned} \min_{\Theta} \mathcal{L} =& -\log(p(\mathbf{Y}|\Theta, \mathbf{G}_u, \mathbf{G}_v) \cdot p(\mathbf{G}_u, \mathbf{G}_v|\Theta) \cdot p(\Theta)) \\ =& \sum_{y_{uv} \in \mathbf{Y}} -(y_{uv} \log \hat{y}_{uv} + (1 - y_{uv}) \log(1 - \hat{y}_{uv})) + \lambda_r \|\Theta\|_2^2 \\ & - \lambda_u \sum_{r_u \in \mathcal{R}_u} \|\mathbf{I}_{r_u} - \mathbf{E}_u^\top \tilde{\mathbf{r}}_u \mathbf{E}_u\|_2^2 - \lambda_v \sum_{r_v \in \mathcal{R}_v} \|\mathbf{I}_{r_v} - \mathbf{E}_v^\top \tilde{\mathbf{r}}_v \mathbf{E}_v\|_2^2, \end{aligned} \tag{19}$$

where $\mathbf{I}_{r_u}$ is the slice of the indicator tensor $\mathbf{I}_{R_u}$ in the user relational graph for relation r_u, and $\mathbf{I}_{r_v}$ is the slice of the indicator tensor $\mathbf{I}_{R_v}$ in the item relational graph for relation r_v. Embedding matrices $\tilde{\mathbf{r}}_v$ and $\tilde{\mathbf{r}}_v$ correspond to the user relation r_u and item relation r_v, respectively. λ_u, λ_v are balancing hyper-parameters. Note that we also add regularization term $\|\Theta\|_2^2$ to control the model complexity, and λ_r is the coefficient of the regularization term. Algorithm 2 shows the training process of the framework.

Algorithm 2: Recommendation framework's training process

Input : Interaction matrix $\mathbf{Y}$; item relational graph $\mathbf{G}_v$; user relational graph $\mathbf{G}_u$; trainable parameters: $\Theta = \{\mathbf{E} = [\mathbf{E}_u, \mathbf{E}_v], \mathbf{R} = [\mathbf{R}_u, \mathbf{R}_v]\}$; hyper-parameters: d, H, η, λ_u, λ_v, λ_r.

Output: Prediction function $\mathcal{F}(u, v|\Theta, \mathbf{Y}, \mathbf{G}_u, \mathbf{G}_v)$.

```
Initialize all the parameters in Θ;
for number of iterations do
    Draw a batch of positive and negative interaction data from Y;
    Draw a batch of true and false triples from G_u;
    Draw a batch of true and false triples from G_v;
    Calculate loss L according to Equation (19) ;
    for each parameter θ ∈ Θ do
        Compute ∂L/∂θ on the minibatch by the back-propagation;
        Update θ ← θ − η · ∂L/∂θ.
    end
end
Return F(u, v|Θ, Y, G_u, G_v).
```

3. Discussion

In this section, we first present the time complexity analysis of our IAGCF and then compare IAGCF with some popular collaborative filtering methods.

3.1. Time complexity analysis

Let M be the number of users, N be the number of items, $|\mathbf{Y}|$ be the number of user–item interactions, K be the number of attributes, L_i be the number of attribute a_i's values, T be the number of neighbors for relational graphs, H be the number of convolution layers, d be the embedding size. In what follows, we analyze the time complexity of IAGCF. We illustrate IAGCF's time complexity from two aspects, i.e., the graph construction and recommendation framework. (i) The computational cost for the graph construction primarily arises from the construction of strength matrices and the construction of user and item relational graphs. Specifically, for constructing the strength matrix for attribute a_i, the complexity is $O((M + N) \cdot L_i)$. For the relational graph construction, the complexity is $O(M)$ and $O(N)$ for a user and an item. It is worth noting that acceleration techniques, as discussed in Bentley (1975) and Liu et al. (2006), can be employed to expedite the graph construction process. Importantly, the graph construction can be completed offline prior to the training phase of the recommendation framework. (ii) The computational cost for the recommendation framework is mainly derived from the aggregation layer, the prediction layer, and the computation of the likelihood for relational graphs. For a user and an item, the complexity of the aggregation layer is $O(K \cdot T \cdot d \cdot H)$. The complexity of the prediction layer is $O(|\mathbf{Y}| \cdot d)$. The complexity for computing the likelihood of relational graphs is $O(K \cdot T \cdot d^2)$. In general, the complexity of each training iteration of the recommendation framework is $O(K \cdot T \cdot (|\mathbf{Y}| \cdot d \cdot H + d^2))$.

3.2. Comparisons to popular collaborative filtering based models

Collaborative Filtering (CF) operates on the heuristic that users with similar historical behaviors likely share comparable item preferences (Chou et al., 2022; Hu et al., 2008; Wang, Wu et al., 2022). In this context, matrix factorization (MF) has become a dominant technique. It represents users and items in a common latent space and models user preferences via the inner product of corresponding user and item embeddings (Koren et al., 2009). However, the simplistic representations in MF-based methods may not effectively capture intricate user–item interactions. In recent times, deep neural network techniques have shown marked success in diverse research domains like computer vision and natural language processing. Some innovative recommender systems (He et al., 2017; Xue et al., 2017) employ these networks to better understand complex user–item relations, thus outperforming earlier, more simplistic models. Yet, a notable limitation of CF methods is their struggle with sparse data — situations where limited interactions exist between users and items. Many methods circumvent this by integrating auxiliary information, such as item attributes. This auxiliary data is typically transformed into a feature vector combined with user and item IDs and is then used in a supervised learning model to predict scores. This methodology has found substantial applications in industry (Guo et al., 2017; Huang et al., 2019; Rendle, 2010). However, these models often treat each interaction in isolation, neglecting interconnected user and item relationships. While some strategies leverage graph convolutions on user–item bipartite graphs to encapsulate higher-order interaction relations (Berg et al., 2018; Liu et al., 2021; Wang, He, Wang et al., 2019), they primarily consider collaborative signals (i.e., interactions reflected in a user's history). It is essential to acknowledge that in real-world settings, various semantic similarities among users and items exist, supplementing the collaborative information. Addressing this, our paper introduces a new CF technique named IAGCF. Contrary to previously discussed models, IAGCF harnesses item attributes to define relations of similarity among users and items. Following this, the model aggregates these relational neighborhoods to bolster the representations of users and items. The experimental results confirm that IAGCF offers superior recommendation performance compared to the above methodologies.

4. Experiments

In this section, we first introduce the experimental settings. Then, we compare our IAGCF with other recommenders to show its effectiveness. Next, we conduct a detailed model analysis to study the components of IAGCF. Finally, we study the impact of the hyper-parameters in IAGCF.

4.1. Experimental setup

In what follows, we introduce four aspects: datasets, compared recommender algorithms, evaluation metrics, and the configuration of hyper-parameters.

4.1.1. Dataset description

We adopt six real-world datasets: Amazon,[1] DBook,[2] DMovie,[3] Yelp,[4] KKBox,[5] and IMDB.[6] Each dataset contains not only the user–item interactions, but also the attribute information of items. The basic information (i.e., the user–item interactions, item attributes, and application scenarios) of these datasets is summarized in Table 2.

[1] Amazon: http://jmcauley.ucsd.edu/data/amazon/.
[2] DBook: http://book.douban.com.
[3] DMovie: http://movie.douban.com.
[4] Yelp: https://www.yelp.com/.
[5] KKBox: https://wsdm-cup-2018.kkbox.events/.
[6] IMDB: https://www.imdb.com/.

4.1.2. Compared models

We compare IAGCF with 12 models, which can be divided into three classes: (i) CF models (i.e., MF Koren et al., 2009, NeuCF He et al., 2017, GCMC Berg et al., 2018, and DGCF Liu et al., 2021), (ii) feature-enhanced models (i.e., FM Rendle, 2010, DeepFM Guo et al., 2017, FiBiNET Huang et al., 2019, and MKR Wang, Zhang, Zhao et al., 2019), and (iii) heterogeneous graph convolution models (i.e., KGAT Wang, He et al., 2019, KGCN Wang, Zhao et al., 2019, KGNN-LS Wang, Zhang, Zhang et al., 2019, and KLGCN Wang, Li et al., 2022). Table 3 concludes the characteristics of them. The brief descriptions for these algorithms are listed as follows:

CF models: These approaches leverage the interaction relations for recommendation. To incorporate item feature information into CF models, we concatenate feature embeddings and ID embeddings as the item embeddings.

- MF is a classical baseline, which uses the inner product between user and item embeddings to model interactions (Koren et al., 2009).
- NeuCF is a deep learning-based CF model, which combines a generalized MF framework with a multi-layer perceptron to model user–item interactions (He et al., 2017).
- GCMC exploits a graph auto-encoder on the user–item interaction graph to use the direct user–item connection (Berg et al., 2018).
- DGCF is a graph-based method, which proposes a dual-message propagation mechanism to consider both collaborative signals and co-occurrence relations (Liu et al., 2021).

Feature interaction models: These approaches consider combinatorial features by modeling the feature interactions.

- FM is a feature-enhanced method, which models the second-order feature interaction (Rendle, 2010).
- DeepFM improves the FM model by deep learning methods. It uses a shallow component and a deep component to model the complex feature interaction (Guo et al., 2017).
- FiBiNET is a deep factorization machine model, which introduces the squeeze-and-excitation network mechanism model the feature importance (Huang et al., 2019).
- MKR is an end-to-end framework leveraging knowledge graph embeddings to enhance recommendation by sharing latent features between items and knowledge graph entities (Wang, Zhang, Zhao et al., 2019).

Heterogeneous graph convolution models: These approaches utilize various graph neural networks to consider different relationships between entities (i.e., users, items, and attributes) and model entity embeddings.

- KGAT leverages both the knowledge graph embedding method and graph convolution network to learn the entity representations for recommendation (Wang, He et al., 2019).
- KGCN extends graph convolution network algorithms to item knowledge graphs for knowledge-based recommendation tasks. It considers item semantic neighborhoods in the graph to enhance item representations (Wang, Zhao et al., 2019).
- KGNN-LS employs graph neural networks and label smoothness regularization for personalized item recommendations (Wang, Zhang, Zhang et al., 2019).
- KLGCN enhances the KGCN model by removing linear transformation and activation. It also considers collaborative signals from the user–item interaction graph (Wang, Li et al., 2022).

4.1.3. Evaluation metric

In the experiment, two tasks are considered to assess the effectiveness of IAGCF: click-through rate (CTR) prediction and top-K recommendation. For the first task, we use Area Under Curve (AUC) and

Table 2
Statistical details of the six datasets.

Dataset	# users	# items	# interactions	Density	Attributes of items	Scenario
Amazon	6145	2753	97,891	0.5786%	brand, category	Product
DBook	12,546	22,179	430,466	0.1547%	author, publisher	Book
DMovie	12,896	12,010	861,304	0.5561%	genre, actor, director	Movie
Yelp	48,866	28,596	316,684	0.0227%	city, category	Business
KKBox	20,816	19,724	1,512,678	0.3684%	genre, lyricist, composer, artist	Music
IMDB	2112	9917	690,151	3.2951%	genre, actor, director, country, location, tag	Movie

Table 3
The characteristics of methods. 'inter.-data' and 'att.-data' indicate whether to utilize user–item interaction data and item attribute data, respectively; 'c-relation', 'u-relation', and 'i-relation' indicate whether to consider collaborative signals, user similarity relations based on item attributes, and item similarity relations based on item attributes, respectively; 'n-comp' and 'g-comp' indicate whether to use neural components and graph convolution components, respectively.

Method	inter.-data	att.-data	c-relation	u-relation	i-relation	n-comp.	g-comp.
MF	✓						
NeuCF	✓					✓	
GCMC	✓		✓			✓	✓
DGCF	✓		✓			✓	✓
FM	✓	✓					
DeepFM	✓	✓				✓	
FiBiNET	✓	✓				✓	
MKR	✓	✓				✓	
KGAT	✓	✓	✓			✓	✓
KGCN	✓	✓				✓	✓
KGNN-LS	✓	✓				✓	✓
KLGCN	✓	✓	✓			✓	✓
IAGCF	✓	✓	✓	✓	✓	✓	✓

F1 as the metrics. These metrics are widely adopted in the CTR prediction task. For the top-K recommendation, we utilize three metrics: Precision@K (P@K), Recall@K (R@K) and F1@K (F@K). We follow the ways in He et al. (2017), Li and Yang (2021) and Wu, Li et al. (2020) to reduce the computational cost in the ranking process. Specifically, for each user, we sample 2000 negative items and combine these items with positive items in ranking. We set K = 10 by default.

4.1.4. Hyper-parameter settings

The datasets are randomly divided into three parts with the ratio 6:2:2 for training, testing, and validation, respectively. We utilize the Adam optimizer to estimate parameters for all methods. The batch size is tuned amongst {128, 256, 512, 1024, 2048} and the learning rate is tuned from {0.001, 0.005, 0.01, 0.05, 0.1}. In addition, the embedding size is tested in {4, 8, 16, 32, 64} for all models. For our IAGCF, we set the user and item similar neighbor size T as 5 and the aggregation layer number H as 1. Moreover, the balancing factors λ_u, λ_v are chosen from {0.0001, 0.001, 0.01, 0.1}. For the similarity strength calculation, we unified use the cosine similarity function $\mathcal{S}_c(\cdot,\cdot)$ in graph construction by default. We will study these hyper-parameters for IAGCF in the following experiments. Some key parameter settings in the baselines are introduced below. The number of hidden layers (i.e., the multi-layer perceptron in DeepFM, FiBiNET, and MKR, and graph convolution operations in KGAT, KGCN, KGNN-LS, and KLGCN) is tuned amongst {1, 2, 3, 4}. The neighbor sampling size for KGCN is tested in {4, 8, 16, 32, 64}. The balancing hyper-parameters ω and μ in KLGCN is as $\omega = \mu = 0.5$. Some other hyper-parameter settings are following their original papers.

4.2. Performance comparison

Table 4 reports the comparison results of the CTR prediction task and top-K recommendation, respectively. The main observations are summarized as follows:

(i) For CF methods, MF shows poor performance, indicating inner products between embeddings are not sufficient for learning user–item interactions. NeuCF generally performs better than MF, which shows the effectiveness of neural architectures in feature modeling. Graph-based recommenders achieve remarkable improvements, which indicates the positive effects of considering the neighborhood information. DGCF generally achieves better performance than GCMC. These results show that considering user-user and item–item relations explicitly can improve recommendation.

(ii) Both MF and FM are shallow approaches, while FM generally outperforms MF. For deep recommenders, DeepFM and FiBiNet generally yield better performance than NeuCF. These results indicate that feature interactions are effective means in recommendation. Moreover, FM underperforms DeepFM and FiBiNet in most cases, which shows that deep components are expressive for feature interaction approaches.

(iii) Heterogeneous graph convolution models achieve quite strong performance among baselines, verifying the importance of modeling the higher-order relations between entities in the heterogeneous graph. Moreover, since KLGCN and KGAT further consider collaborative signals from the user–item bipartite graph, these two methods generally achieve better performance than KGCN, which is consistent with the recent study (Wang, Li et al., 2022).

(iv) Our IAGCF outperforms other recommenders in both CTR prediction and top-K recommendation, which indicates that leveraging item attribute-aware similarities of users and items can improve the performance.

4.3. Performance comparison in sparse scenarios

The problem of sparse interactions is one of the challenges that recommender systems face. In this subsection, we study the performance of IAGCF in sparse data scenarios. Specifically, we utilize x% user–item interactions from the training data, and keep the test set and validation set fixed. In this way, we generate different sparsity-level data for experiments. For comparison, we selected three representative baselines (i.e., DGCF, FiBiNet, and KLGCN). The AUC results on six datasets are reported in Figs. 6 and 7, where ratio x is varied as $\{25, 50, 75\}$. From the results, we find that (i) The AUC results of four approaches deteriorate with the increase of the data sparsity level. (ii) Compared with other methods, IAGCF shows the best performance in different sparse scenarios, which shows that IAGCF is more robust. Therefore, leveraging user and item similarity relations is capable to alleviate data sparsity issues.

Table 4
Comparison results with baselines in the CTR prediction and top-K recommendation tasks.

Method	Amazon					DBook				
	AUC	F1	P@10	R@10	F@10	AUC	F1	P@10	R@10	F@10
MF	0.6426	0.6008	0.0096	0.0280	0.0143	0.7607	0.6828	0.1103	0.1675	0.1330
NeuCF	0.6607	0.6075	0.0115	0.0333	0.0171	0.7780	0.6911	0.1090	0.1642	0.1310
GCMC	0.6762	0.6089	0.0103	0.0287	0.0152	0.7643	0.6893	0.1026	0.1617	0.1255
DGCF	0.6859	0.6124	0.0112	0.0315	0.0165	0.7732	0.7004	0.1055	0.1663	0.1291
FM	0.6570	0.6137	0.0109	0.0308	0.0161	0.8026	0.7221	0.1089	0.1656	0.1314
DeepFM	0.6714	0.6376	0.0117	0.0338	0.0174	0.7891	0.7079	0.1188	0.1802	0.1432
FiBiNet	0.6913	0.6460	0.0120	0.0354	0.0179	0.8113	0.7426	0.1127	0.1708	0.1358
MKR	0.6825	0.6293	0.0107	0.0321	0.0160	0.8074	0.7338	0.1143	0.1720	0.1373
KGAT	0.6935	0.6477	0.0134	0.0359	0.0195	0.7995	0.7297	0.1152	0.1791	0.1402
KGCN	0.6868	0.6451	0.0122	0.0330	0.0178	0.7865	0.7132	0.1206	0.1787	0.1440
KGNN-LS	0.6860	0.6442	0.0118	0.0320	0.0172	0.7871	0.7188	0.1175	0.1780	0.1416
KLGCN	0.7041	0.6648	0.0141	0.0383	0.0206	0.8079	0.7363	0.1241	0.1869	0.1492
IAGCF	**0.7273**	**0.6891**	**0.0158**	**0.0432**	**0.0231**	**0.8211**	**0.7450**	**0.1280**	**0.1927**	**0.1538**
Method	DMovie					Yelp				
	AUC	F1	P@10	R@10	F@10	AUC	F1	P@10	R@10	F@10
MF	0.8596	0.7870	0.2140	0.1897	0.2011	0.8206	0.7413	0.0341	0.1678	0.0567
NeuCF	0.8709	0.7996	0.2176	0.1943	0.2053	0.8154	0.7367	0.0326	0.1550	0.0539
GCMC	0.8620	0.7884	0.2155	0.1839	0.1984	0.8358	0.7570	0.0338	0.1592	0.0558
DGCF	0.8683	0.7921	0.2201	0.1929	0.2056	0.8528	0.7730	0.0352	0.1656	0.0581
FM	0.8715	0.8009	0.2226	0.2008	0.2111	0.8283	0.7442	0.0374	0.1834	0.0621
DeepFM	0.8836	0.8107	0.2254	0.2032	0.2137	0.8440	0.7694	0.0370	0.1848	0.0617
FiBiNet	0.8809	0.8096	0.2237	0.2005	0.2115	0.8495	0.7806	0.0385	0.1899	0.0640
MKR	0.8691	0.7943	0.2205	0.1923	0.2054	0.8532	0.7780	0.0381	0.1870	0.0633
KGAT	0.8912	0.8218	0.2252	0.1974	0.2104	0.8561	0.7768	0.0365	0.1731	0.0603
KGCN	0.8757	0.8039	0.2213	0.1910	0.2050	0.8582	0.7785	0.0377	0.1802	0.0624
KGNN-LS	0.8820	0.8105	0.2221	0.1937	0.2069	0.8591	0.7799	0.0378	0.1805	0.0625
KLGCN	0.8874	0.8153	0.2268	0.1966	0.2106	0.8665	0.7854	0.0403	0.1927	0.0667
IAGCF	**0.8988**	**0.8245**	**0.2308**	**0.2170**	**0.2237**	**0.8776**	**0.7861**	**0.0436**	**0.2084**	**0.0721**
Method	KKBox					IMDB				
	AUC	F1	P@10	R@10	F@10	AUC	F1	P@10	R@10	F@10
MF	0.8779	0.8052	0.2761	0.2298	0.2508	0.8617	0.7886	0.4247	0.1103	0.1751
NeuCF	0.8908	0.8124	0.2857	0.2271	0.2531	0.8836	0.8094	0.4315	0.1098	0.1750
GCMC	0.8955	0.8185	0.2920	0.2447	0.2663	0.8770	0.8038	0.4302	0.1079	0.1725
DGCF	0.9107	0.8348	0.2976	0.2493	0.2713	0.8744	0.8015	0.4340	0.1122	0.1783
FM	0.8896	0.8139	0.2887	0.2422	0.2634	0.8902	0.8163	0.4434	0.1149	0.1825
DeepFM	0.9093	0.8311	0.2990	0.2506	0.2727	0.8873	0.8145	0.4389	0.1110	0.1772
FiBiNet	0.9141	0.8378	0.3063	0.2583	0.2803	0.8920	0.8171	0.4466	0.1135	0.1810
MKR	0.9124	0.8344	0.3015	0.2528	0.2750	0.8856	0.8102	0.4413	0.1120	0.1787
KGAT	0.9010	0.8307	0.2939	0.2467	0.2682	0.8898	0.8160	0.4403	0.1133	0.1802
KGCN	0.8969	0.8202	0.2995	0.2501	0.2725	0.8861	0.8159	0.4355	0.1126	0.1789
KGNN-LS	0.8982	0.8273	0.3022	0.2502	0.2738	0.8865	0.8177	0.4395	0.1131	0.1799
KLGCN	0.9027	0.8296	0.3018	0.2524	0.2749	0.8872	0.8127	0.4411	0.1145	0.1818
IAGCF	**0.9235**	**0.8516**	**0.3169**	**0.2740**	**0.2939**	**0.8955**	**0.8192**	**0.4552**	**0.1204**	**0.1904**

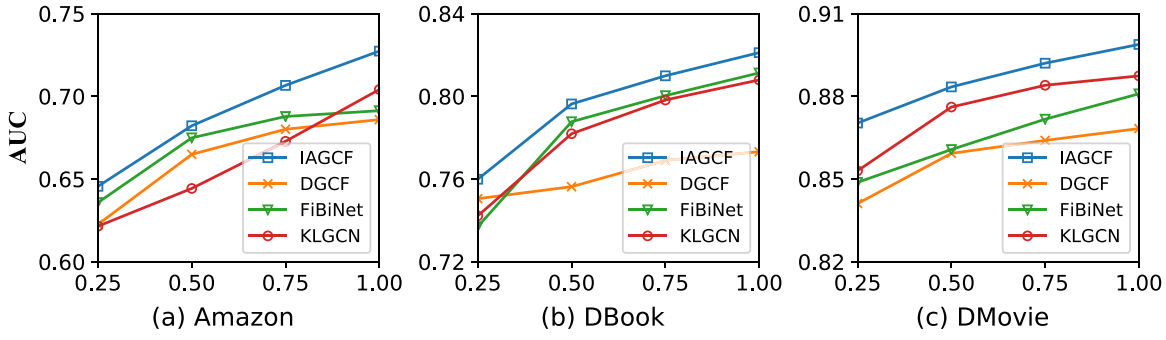


Fig. 6. Performance comparison of three representative models and our IAGCF w.r.t. different ratios of training data on Amazon, DBook, and DMovie datasets.

4.4. Ablation study

We perform ablation experiments for the fine-grained analysis of our IAGCF. Specifically, we study the effect of the attention mechanism in IAGCF. In the recommendation framework of IAGCF, we design an aggregator to aggregate the neighborhoods. Specifically, we propose an attention mechanism to consider the relations. In what follows, we conduct ablation experiments to study the effect of the attention mechanism. We design a variant IAGCF-M, which utilizes the mean pooling to aggregate the neighborhoods. The AUC results of the variant are reported in Figs. 8 and 9, where FiBiNet and KLGCN are added as baselines. From the results, we find that (i) IAGCF shows consistent improvements over IAGCF-M, which indicates that the attention mechanism can provide the importance of neighbors in different relationships so as to enhance the performance. (ii) Although the attention mechanism is not used, IAGCF-M still performs better than other baselines,

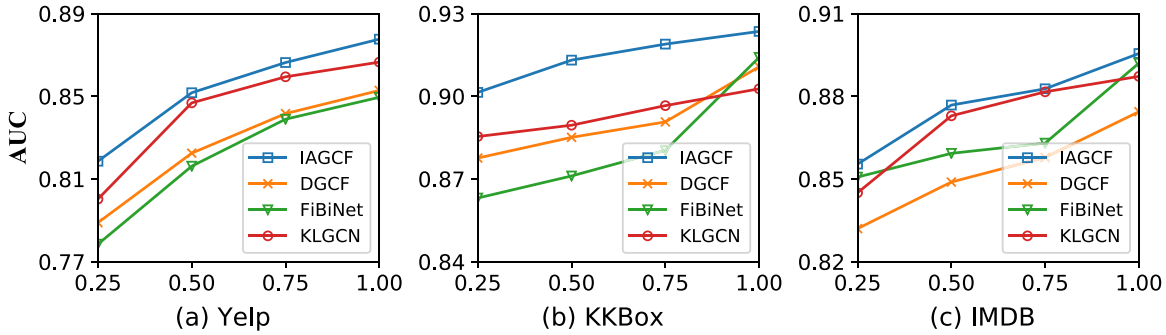


Fig. 7. Performance comparison of three representative models and our IAGCF w.r.t. different ratios of training data on Yelp, KKBox, and IMDB datasets.

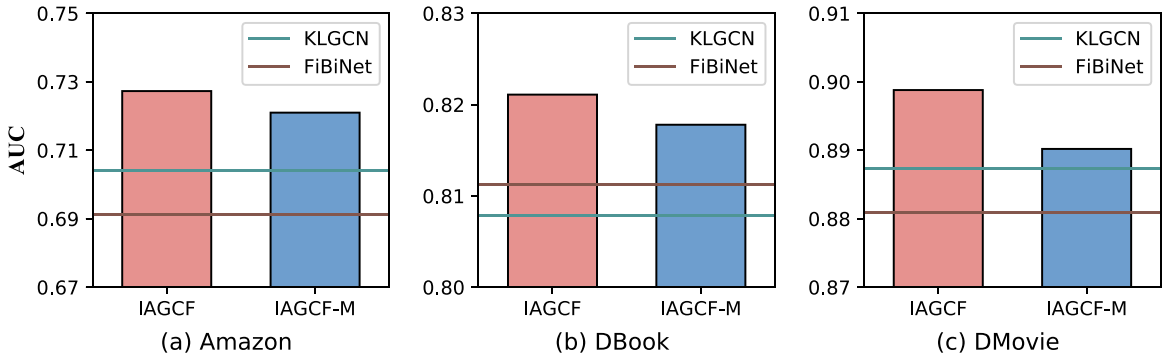


Fig. 8. Effect of the attention mechanism in IAGCF on Amazon, DBook, and DMovie datasets. Two lines show the results of FiBiNet and KLGCN, respectively, which are added as baselines.

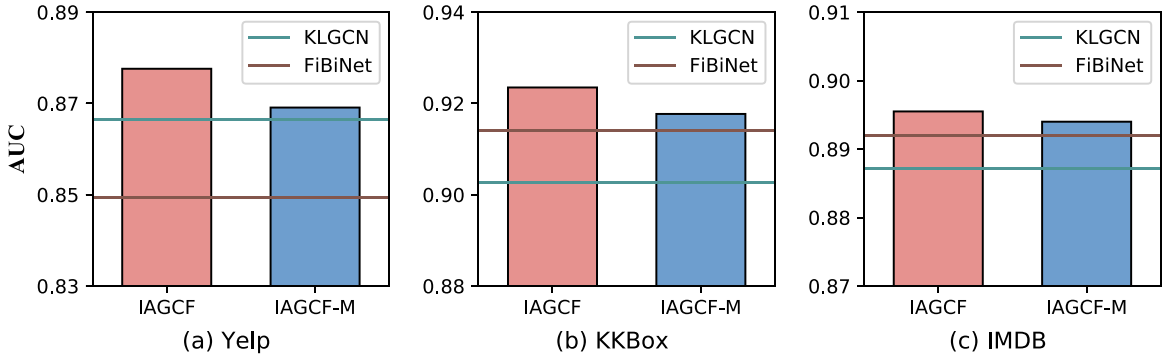


Fig. 9. Effect of the attention mechanism in IAGCF on Yelp, KKBox, and IMDB datasets. Two lines show the results of FiBiNet and KLGCN, respectively, which are added as baselines.

which shows that our method can provide high-quality user and item similar neighbors for aggregation.

4.5. Hyper-parameter sensitivity

In this subsection, we study four aspects hyper-parameters (i.e., the embedding size d, the aggregation layer number H, the neighbor size T, and the balancing factors λ_u, λ_v) to conduct detailed analysis of our IAGCF.

4.5.1. Embedding size

We show the impact of embedding size d on AUC/F1 results in Figs. 10 and 11. We find that IAGCF performs well when $d = 8$ or 16 in most cases. Too large embedding size will not enhance the performance, or even reduce the accuracy. This may be caused by the overfitting problem.

4.5.2. Layer number

The impact of aggregation layer number H on AUC/F1 results is shown in Figs. 12 and 13. We find that properly increasing the number of layers can improve performance. However, too large layer number will increase the time complexity. Thus, a proper layer size is needed to balance the trade-off between accuracy and time cost. In general, two-hop is a good choice in our experiments.

4.5.3. Neighbor size

In IAGCF, we use top-T similarity strength users/items to be the neighborhood for each user/item. We study IAGCF's performance with neighbor size T in Figs. 14 and 15. From the results, we find that the accuracy first increases and then begins to decline. One possible reason is that too many neighbors may introduce irrelevant information, which will degrade recommendation performance.

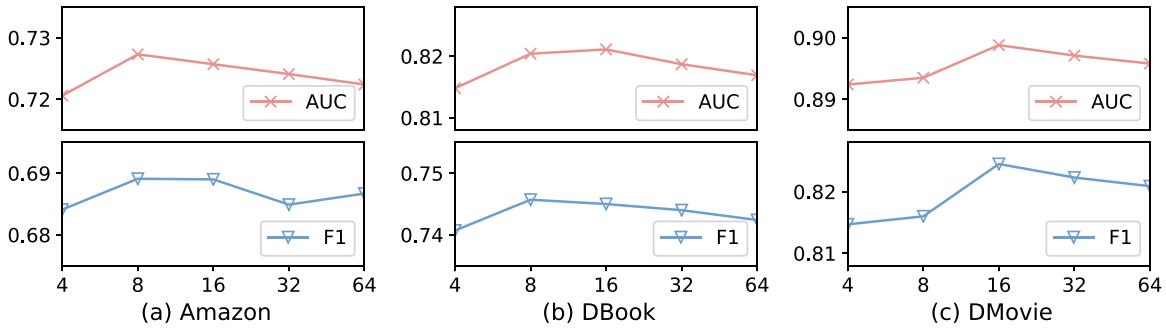


Fig. 10. Effect of the embedding size d in IAGCF on Amazon, DBook, and DMovie datasets.

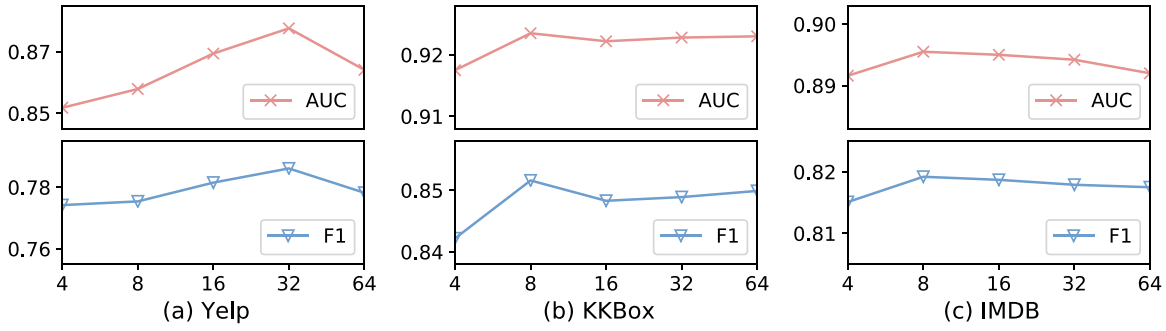


Fig. 11. Effect of the embedding size d in IAGCF on Yelp, KKBox, and IMDB datasets.

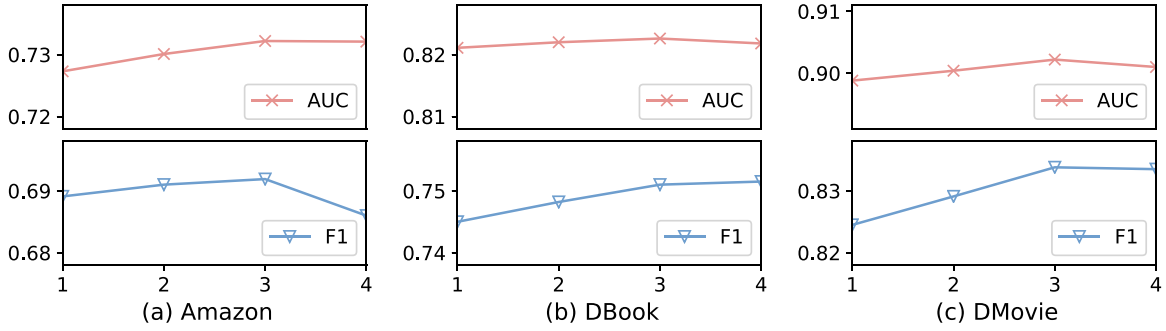


Fig. 12. Effect of the aggregation layer number H in IAGCF on Amazon, DBook, and DMovie datasets.

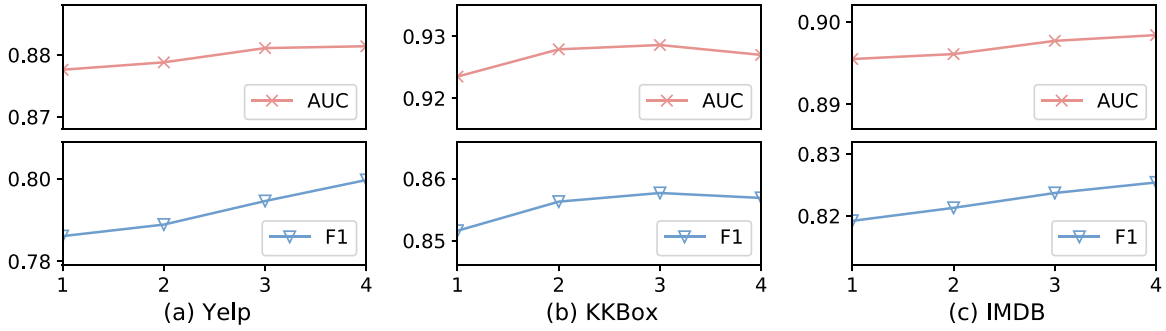


Fig. 13. Effect of the aggregation layer number H in IAGCF on Yelp, KKBox, and IMDB datasets.

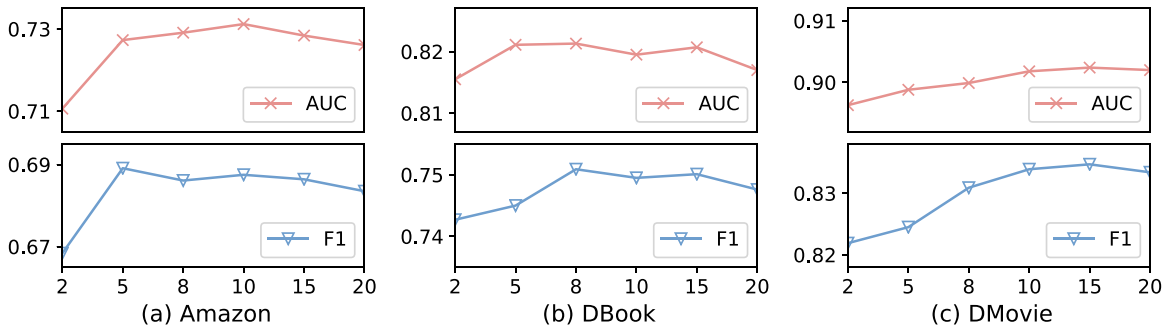


Fig. 14. Effect of the neighbor size T in IAGCF on Amazon, DBook, and DMovie datasets.

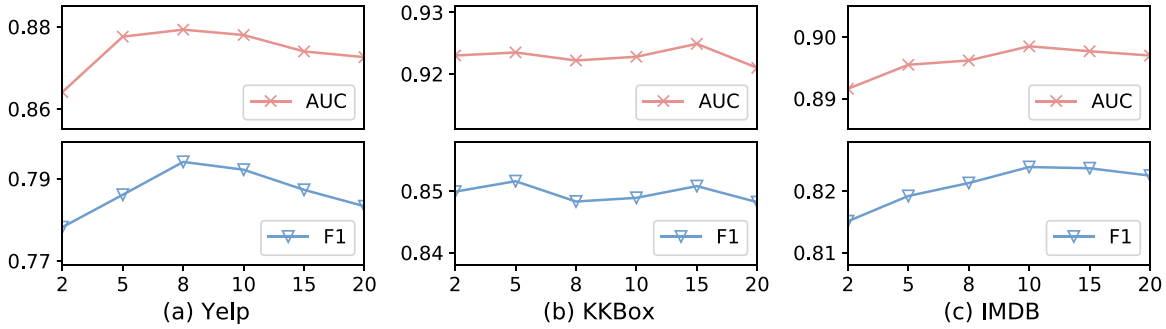


Fig. 15. Effect of the neighbor size T in IAGCF on Yelp, KKBox, and IMDB datasets.

4.5.4. Balancing factors

Balancing factors λ_u, λ_v are key hyper-parameters for model learning. The impact of λ_u, λ_v on AUC results is shown in Figs. 16 and 17. Generally, when λ_u, λ_v = 0.01 or 0.001, IAGCF achieves a nice performance. We analyze that too small weight may not effectively constrain the item attribute-aware similar relations of users and items, while too large weight may mislead the recommendation task.

5. Related work

In this section, we introduce three research directions that are relevant to our work, i.e., collaborative filtering, relation-aware recommendation, and graph-based recommendation.

5.1. Collaborative filtering

Collaborative Filtering (CF), whose heuristic is that users who have resemble historical behaviors tend to share similar preferences for items, provides an effective way to learn users' preferences for items (Alharbe et al., 2023; Chou et al., 2022; Hu et al., 2008; Li, Yang, Huo and Hussain, 2022; Wang, Wu et al., 2022). How to model informative representations for users and items is the key to CF (Wu et al., 2022). Early CF models, such as matrix factorization (MF), use free embeddings to represent users and items, and exploit the inner product to learn interactive behaviors (Koren et al., 2009). Recently, with the success of deep learning (Zhang et al., 2020b, 2022), researchers use a variety of neural components to design the recommenders (Zhang et al., 2019). For example, DMF exploits the multi-layer perceptron to model user and item features (Xue et al., 2017). For further enriching the representations, some follow-on works consider leveraging the historical behaviors of users (He et al., 2018; Kabbur et al., 2013; Koren, 2008). FISM aggregates representations of historical items with the mean pooling operation, and utilizes the aggregated representations as user embeddings (Kabbur et al., 2013). NAIS, which enhances the performance of FISM, introduces the attention mechanism to weigh historical items in the aggregation process (He et al., 2018). Since user–item interactions can be represented as a bipartite graph, some recent works utilize graph convolution techniques to inject high-order connections into representation learning (Wu, Sun et al., 2020). For instance, NGCF, which borrows the graph convolutional networks, devises message construction and message aggregation process to perform embedding propagation (Wang, He, Wang et al., 2019). LightGCN further enhances NGCF by removing some redundant operations in graph convolution layers, i.e., activation functions and linear transformations (He et al., 2020). In real scenarios, since attribute information is usually associated with users or items, factorization machines (FMs) are designed to integrate these attributes into the CF tasks (Rendle, 2010). Some follow-on works further utilize the multi-layer perceptron (Guo et al., 2017) and the attention mechanism (Xiao et al., 2017) to enhance the FM model. Another line of study uses attribute information to construct the network schema (e.g., meta-paths). HERec utilizes the meta-path-based random walk to generate user and item features and adopts the matrix factorization framework for recommendation (Shi et al., 2018). NeuACF utilizes deep neural networks to extract aspect-level features for users and items from meta-paths, and design an attention mechanism to fuse different aspect factors (Shi et al., 2019, 2020). In this paper, we present a novel CF approach IAGCF, which leverages item attributes to enhance the embedding modeling. Different from the aforementioned models, our IAGCF exploits item attributes to construct user and item similar relations. Then, IAGCF aggregates similar neighborhoods to enrich the representations for users and items.

5.2. Relation-aware recommendation

Various types of relational data are widely used by recommender systems to improve performance. Some studies mine relationships directly from the user–item interaction data. For example, MCF, which proposes to use also-viewed relations, exploits a matrix co-factorization approach to jointly factorize interaction data and also-viewed data for rating prediction tasks (Park et al., 2017). CMN considers user co-occurrence relations and adopts a memory network to integrate these relations into the recommendation (Ebesu et al., 2018). MMCF devises a multiplex memory network, which models both user and item

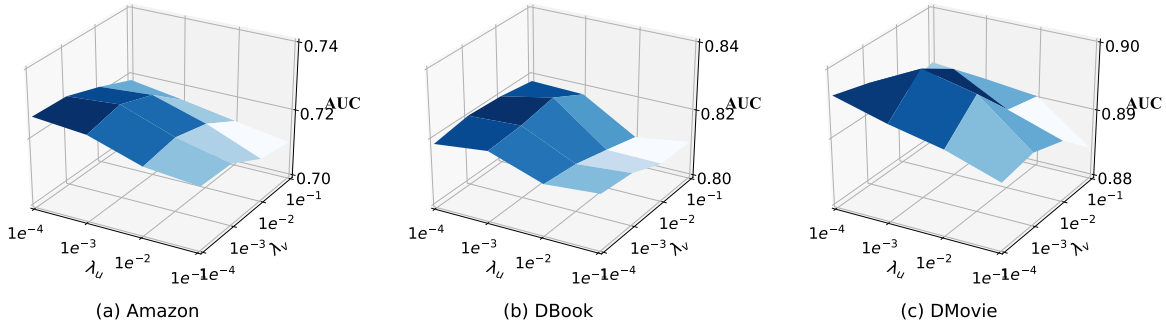


Fig. 16. Effect of the balancing factors λ_u and λ_v in IAGCF on Amazon, DBook, and DMovie datasets.

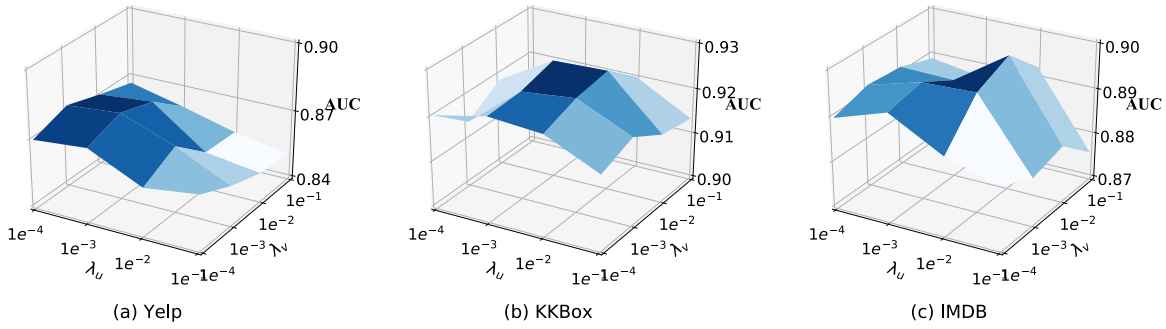


Fig. 17. Effect of the balancing factors λ_u and λ_v in IAGCF on Yelp, KKBox, and IMDB datasets.

co-occurrence contexts, to enhance CMN (Jiang et al., 2020). Based on the co-occurrence relationship, some works define the measure of similarity, and select high-quality co-occurrence neighbors according to the pre-defined threshold (Fu et al., 2021; Wu, Zhang et al., 2019). Another line of recommendation study utilizes the relationships from auxiliary information. Social links are a commonly used relationship to assist recommendation. Early works, such as TrustSVD (Guo et al., 2015) and SoReg (Ma et al., 2011), utilize the MF framework to fuse social information. As deep learning grows, some neural components (e.g., memory network Chen, Zhang, Liu et al., 2019, multi-layer perceptron Feng et al., 2022; Wu, Sun et al., 2018, and attention mechanism Chen, Zhang, Wang et al., 2019; Li, Yang, Hussain et al., 2022) are used in social recommenders. For example, SoNeuMF integrates deep learning to model both user–item and user-user interactions in social recommendation, overcoming matrix factorization's limitations in capturing complex nonlinear relationships (Feng et al., 2022). By treating social links as social graphs, many recent works, such as Diffnet (Wu, Li et al., 2020) and ME-LGNN (Miao et al., 2022), use graph convolutions to model high-order social relationships. Knowledge graph is also typical relational data and is widely utilized in recommender systems. Knowledge-based recommenders can be divided into three types i.e., path-based models (Shi et al., 2018), embedding-based models (Xin et al., 2019), and propagation-based models (Wang, He et al., 2019; Wang, Zhao et al., 2019). These methods use the attribute and structure of the knowledge graph from different aspects and each of them is proven to be effective (Guo et al., 2020). In addition, researchers also use session sequences (Wu, Tang et al., 2019) and textual reviews (Wu, Quan et al., 2018) to construct relations in recommendation. In this paper, we focus on leveraging the user and item similarity for recommendation. Specifically, we propose a relation construction strategy to find similar relations based on the items' attributes.

5.3. Graph representation

Graph representation learning has attracted extensive attention in academia and industry. The main purpose of graph representation learning is to learn node low-dimensional embeddings while capturing the information from the graph (e.g., node content and topology structure) (Liu et al., 2023; Shi et al., 2022; Yan et al., 2023; Zang et al., 2021; Zhang et al., 2018; Zhu et al., 2023). Generally speaking, graph representation learning includes two mainstream types of approaches: unsupervised and semi-supervised approaches. Most unsupervised methods aim to preserve graph topology for node embedding learning. For instance, DeepWalk utilizes skip-gram and random walk for representation learning (Perozzi et al., 2014). Node2vec further extends the idea of DeepWalk by using a biased second-order random walk (Grover & Leskovec, 2016). SDNE exploits deep learning techniques (i.e., the autoencoder model) to capture the highly non-linear network structure (Wang et al., 2016). Struc2vec utilizes topology metrics to construct a multilayered graph and adopts the random walk on the constructed graph for embedding learning (Ribeiro et al., 2017). For the semi-supervised graph representation algorithms, they leverage some labeled nodes to learn embeddings. For instance, GCN devises a message-passing approach, which learns center node embeddings by aggregating the representations from neighborhoods, for the node classification task (Kipf & Welling, 2017). GAT enhances GCN by introducing the attention operation in the neighborhood aggregation (Veličković et al., 2018). CNIM-GCN explicitly preserves common information between feature and topology spaces using a consensus graph, and employs multi-channel graph convolution along with structural and reconstruction consistency constraints (Zhu et al., 2023). Some follow-on works further utilize other techniques such as contrastive learning (Qiu et al., 2020), network geometry (Zhao et al., 2021), and reinforcement learning (Peng et al., 2021) to improve the representation ability. In this paper, we apply graph representation learning to the recommendation tasks. To be specific, a dual graph neural network is designed to aggregate item attribute-aware similar neighborhoods of users and items for better embedding learning.

6. Conclusion and future work

In this work, we propose a recommender IAGCF which considers user and item similarities based on item attributes. Specifically, our

IAGCF first exploits item attributes to find similar neighborhoods for users and items, respectively. Then, we introduce a dual graph neural network to integrate constructed neighborhoods for recommendation. Empirically, we conducted extensive experiments on six datasets. The results show the effectiveness of IAGCF.

In future work, we plan to extend our IAGCF in two directions. First, our current model is primarily designed based on item attributes. We aim to integrate user social information (Ntalianis & Doulamis, 2016; Ntalianis et al., 2017; Stai et al., 2018; Zhao et al., 2017) into the model while continuing to consider item attributes. We hypothesize that simultaneously utilizing both item attributes and user social relations will lead to further improvements in performance. Second, our current model neglects to incorporate prior knowledge information. In response to this, we intend to incorporate prior information, such as logic rules (Zhang et al., 2020a), into our model. This integration is anticipated to enhance the model's effectiveness while also increasing its interpretability.

CRediT authorship contribution statement

Anchen Li: Conceptualization, Methodology, Data curation, Formal analysis, Writing – original draft. **Xueyan Liu:** Writing – review & editing. **Bo Yang:** Methodology, Writing – review & editing, Funding acquisition, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgments

The authors thank all the reviewers for their helpful comments. This work was supported by the National Key R&D Program of China (Grant Nos. 2021ZD0112501 and 2021ZD0112502); the National Natural Science Foundation of China (Grant Nos. U22A2098, 62172185, 62202200 and 62206105).

References

Alharbe, N., Rakrouki, M. A., & Aljohani, A. (2023). A collaborative filtering recommendation algorithm based on embedding representation. *Expert Systems with Applications*, *215*, Article 119380.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), 509–517.

Berg, R. v. d., Kipf, T. N., & Welling, M. (2018). Graph convolutional matrix completion. In *KDD*.

Chen, L., Wu, L., Hong, R., Zhang, K., & Wang, M. (2020). Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *AAAI* (pp. 27–34).

Chen, C., Zhang, M., Liu, Y., & Ma, S. (2019). Social attentional memory network: Modeling aspect-and friend-level differences in recommendation. In *WSDM* (pp. 177–185).

Chen, C., Zhang, M., Wang, C., Ma, W., Li, M., Liu, Y., & Ma, S. (2019). An efficient adaptive transfer neural network for social-aware recommendation. In *SIGIR* (pp. 225–234).

Chou, Y.-C., Chen, C.-T., & Huang, S.-H. (2022). Modeling behavior sequence for personalized fund recommendation with graphical deep collaborative filtering. *Expert Systems with Applications*, *192*, Article 116311.

Ebesu, T., Shen, B., & Fang, Y. (2018). Collaborative memory network for recommendation systems. In *SIGIR* (pp. 515–524).

Feng, X., Liu, Z., Wu, W., & Zuo, W. (2022). Social recommendation via deep neural network-based multi-task learning. *Expert Systems with Applications*, *206*, Article 117755.

Fu, B., Zhang, W., Hu, G., Dai, X., Huang, S., & Chen, J. (2021). Dual side deep context-aware modulation for social recommendation. In *WWW* (pp. 2524–2534).

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *AISTATS* (pp. 249–256). JMLR Workshop and Conference Proceedings.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *KDD* (pp. 855–864).

Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI* (pp. 1725–1731).

Guo, G., Zhang, J., & Yorke-Smith, N. (2015). Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *AAAI*.

Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., & He, Q. (2020). A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR* (pp. 639–648).

He, X., He, Z., Song, J., Liu, Z., Jiang, Y.-G., & Chua, T.-S. (2018). Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *30*(12), 2354–2366.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. In *WWW* (pp. 173–182).

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *ICDM* (pp. 263–272).

Huang, T., Zhang, Z., & Zhang, J. (2019). FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *RecSys* (pp. 169–177).

Jiang, X., Hu, B., Fang, Y., & Shi, C. (2020). Multiplex memory network for collaborative filtering. In *SDM* (pp. 91–99).

Kabbur, S., Ning, X., & Karypis, G. (2013). Fism: factored item similarity models for top-n recommender systems. In *KDD* (pp. 659–667).

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR*.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD* (pp. 426–434).

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37.

Li, A., & Yang, B. (2021). GSIRec: Learning with graph side information for recommendation. *World Wide Web*, *24*, 1411–1437.

Li, A., Yang, B., Huo, H., Chen, H., Xu, G., & Wang, Z. (2022). Hyperbolic neural collaborative recommender. *IEEE Transactions on Knowledge and Data Engineering*.

Li, A., Yang, B., Huo, H., & Hussain, F. (2022). Hypercomplex graph collaborative filtering. In *WWW* (pp. 1914–1922).

Li, A., Yang, B., Hussain, F. K., & Huo, H. (2022). Hsr: Hyperbolic social recommender. *Information Sciences*, *585*, 275–288.

Liu, X., Li, X., Fiumara, G., & De Meo, P. (2023). Link prediction approach combined graph neural network with capsule network. *Expert Systems with Applications*, *212*, Article 118737.

Liu, T., Moore, A. W., Gray, A., & Cardie, C. (2006). New algorithms for efficient high-dimensional nonparametric classification. *Journal of Machine Learning Research*, *7*(6).

Liu, H., Yang, B., & Li, D. (2021). Graph collaborative filtering based on dual-message propagation mechanism. *IEEE Transactions on Cybernetics*, *53*(1), 352–364.

Ma, H., Zhou, D., Liu, C., Lyu, M. R., & King, I. (2011). Recommender systems with social regularization. In *WSDM* (pp. 287–296).

Miao, H., Li, A., & Yang, B. (2022). Meta-path enhanced lightweight graph neural network for social recommendation. In *DASFAA* (pp. 134–149).

Ntalianis, K., & Doulamis, N. (2016). An automatic event-complementing human life summarization scheme based on a social computing method over social media content. *Multimedia Tools and Applications*, *75*, 15123–15149.

Ntalianis, K., Doulamis, A. D., Tsapatsoulis, N., & Mastorakis, N. E. (2017). Social relevance feedback based on multimedia content power. *IEEE Transactions on Computational Social Systems*, *5*(1), 109–117.

Park, C., Kim, D., Oh, J., & Yu, H. (2017). Do" also-viewed" products help user rating prediction? In *WWW* (pp. 1113–1122).

Peng, H., Zhang, R., Dou, Y., Yang, R., Zhang, J., & Yu, P. S. (2021). Reinforced neighborhood selection guided multi-relational graph neural networks. *ACM Transactions on Information Systems*, *40*(4), 1–46.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *KDD* (pp. 701–710).

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., & Tang, J. (2020). Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD* (pp. 1150–1160).

Rendle, S. (2010). Factorization machines. In *ICDM* (pp. 995–1000).

Ribeiro, L. F., Saverese, P. H., & Figueiredo, D. R. (2017). struc2vec: Learning node representations from structural identity. In *KDD* (pp. 385–394).

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook*.

Shi, C., Han, X., Song, L., Wang, X., Wang, S., Du, J., & Philip, S. Y. (2019). Deep collaborative filtering with multi-aspect information in heterogeneous networks. *IEEE Transactions on Knowledge and Data Engineering*, *33*(4), 1413–1425.

Shi, C., Hu, B., Zhao, W. X., & Philip, S. Y. (2018). Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *31*(2), 357–370.

Shi, J., Ji, H., Shi, C., Wang, X., Zhang, Z., & Zhou, J. (2020). Heterogeneous graph neural network for recommendation. In *ICML*.

Shi, D., Li, A., & Yang, B. (2022). Counterfactual-guided and curiosity-driven multi-hop reasoning over knowledge graph. In *International conference on database systems for advanced applications* (pp. 171–179). Springer.

Stai, E., Kafetzoglou, S., Tsiropoulou, E. E., & Papavassiliou, S. (2018). A holistic approach for personalization, relevance feedback & recommendation in enriched multimedia content. *Multimedia Tools and Applications*, *77*, 283–326.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. In *ICLR*.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *KDD* (pp. 1225–1234).

Wang, X., He, X., Cao, Y., Liu, M., & Chua, T.-S. (2019). Kgat: Knowledge graph attention network for recommendation. In *KDD* (pp. 950–958).

Wang, X., He, X., Wang, M., Feng, F., & Chua, T.-S. (2019). Neural graph collaborative filtering. In *SIGIR* (pp. 165–174).

Wang, F., Li, Y., Zhang, Y., & Wei, D. (2022). KLGCN: Knowledge graph-aware Light Graph Convolutional Network for recommender systems. *Expert Systems with Applications*, *195*, Article 116513.

Wang, R., Wu, Z., Lou, J., & Jiang, Y. (2022). Attention-based dynamic user modeling and deep collaborative filtering recommendation. *Expert Systems with Applications*, *188*, Article 116036.

Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., & Wang, Z. (2019). Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 968–977).

Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., & Guo, M. (2019). Multi-task feature learning for knowledge graph enhanced recommendation. In *WWW* (pp. 2000–2010).

Wang, H., Zhao, M., Xie, X., Li, W., & Guo, M. (2019). Knowledge graph convolutional networks for recommender systems. In *WWW* (pp. 3307–3313).

Wu, L., He, X., Wang, X., Zhang, K., & Wang, M. (2022). A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Wu, L., Li, J., Sun, P., Hong, R., Ge, Y., & Wang, M. (2020). Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Wu, L., Quan, C., Li, C., & Ji, D. (2018). Parl: Let strangers speak out what you like. In *CIKM* (pp. 677–686).

Wu, Z.-G., Shi, P., Su, H., & Chu, J. (2011). Passivity analysis for discrete-time stochastic Markovian jump neural networks with mixed time delays. *IEEE Transactions on Neural Networks*, *22*(10), 1566–1575.

Wu, L., Sun, P., Hong, R., Ge, Y., & Wang, M. (2018). Collaborative neural social recommendation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *51*(1), 464–476.

Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2020). Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*.

Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., & Tan, T. (2019). Session-based recommendation with graph neural networks. In *AAAI* (pp. 346–353).

Wu, Q., Zhang, H., Gao, X., He, P., Weng, P., Gao, H., & Chen, G. (2019). Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *WWW* (pp. 2091–2102).

Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., & Chua, T.-S. (2017). Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI* (pp. 3119–3125).

Xin, X., He, X., Zhang, Y., Zhang, Y., & Jose, J. (2019). Relational collaborative filtering: Modeling multiple item relations for recommendation. In *SIGIR* (pp. 125–134).

Xue, H.-J., Dai, X., Zhang, J., Huang, S., & Chen, J. (2017). Deep matrix factorization models for recommender systems. In *IJCAI* (pp. 3203–3209).

Yan, X., Yu, X., Hu, S., & Ye, Y. (2023). Mutual Boost Network for attributed graph clustering. *Expert Systems with Applications*, *229*, Article 120479.

Yang, B., Yih, W.-t., He, X., Gao, J., & Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Zang, X., Yang, B., Liu, X., & Li, A. (2021). DNEA: Dynamic network embedding method for anomaly detection. In *KSEM* (pp. 236–248).

Zhang, Y., Chen, X., Yang, Y., Ramamurthy, A., Li, B., Qi, Y., & Song, L. (2020). Efficient probabilistic logic reasoning with graph neural networks. arXiv preprint arXiv:2001.11850.

Zhang, J., Li, Y., Xiao, W., & Zhang, Z. (2020). Non-iterative and fast deep learning: Multilayer extreme learning machines. *Journal of the Franklin Institute*, *357*(13), 8925–8955.

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, *52*(1), 1–38.

Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2018). Network representation learning: A survey. *IEEE Transactions on Big Data*, *6*(1), 3–28.

Zhang, J., Zhao, Y., Shone, F., Li, Z., Frangi, A. F., Xie, S. Q., & Zhang, Z.-Q. (2022). Physics-informed deep learning for musculoskeletal modeling: Predicting muscle forces and joint kinematics from surface EMG. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *31*, 484–493.

Zhao, Y., Qi, J., Liu, Q., & Zhang, R. (2021). Wgcn: Graph convolutional networks with weighted structural features. In *SIGIR* (pp. 624–633).

Zhao, Z., Yang, Q., Lu, H., Weninger, T., Cai, D., He, X., & Zhuang, Y. (2017). Social-aware movie recommendation via multimodal network learning. *IEEE Transactions on Multimedia*, *20*(2), 430–440.

Zhu, X., Li, C., Guo, J., & Dietze, S. (2023). CNIM-GCN: Consensus neighbor interaction-based multi-channel graph convolutional networks. *Expert Systems with Applications*, *226*, Article 120178.