Information Processing and Management 61 (2024) 103680

Contents lists available at ScienceDirect

Information Processing and Management

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

Integrating user short-term intentions and long-term preferences in heterogeneous hypergraph networks for sequential recommendation

Bingqian Liu [a,1], Duantengchuan Li [a,1], Jian Wang [a,*], Zhihao Wang [a,1], Bing Li [a,b,*], Cheng Zeng [a,*]

[a] School of Computer Science, Wuhan University, Wuhan 430072, China
[b] Hubei Luojia Laboratory, Wuhan, 430079, China

ARTICLE INFO

Keywords:
Sequential recommendation
User intentions
User preferences
Hypergraph network

ABSTRACT

Sequential recommendation tries to model the binary correlations among users and items in a sequence to provide accurate recommendations. However, user behaviors are influenced by both their intentions and preferences. Existing sequential recommendation models cannot effectively capture the user's real preferences and intentions just from the interaction data. To tackle this dilemma, we propose IPSRec, an innovative sequential recommendation approach that explicitly mines user intentions and preferences by fusing heterogeneous auxiliary information from multiple sources. IPSRec employs a heterogeneous hypergraph network to fuse and propagate the topological information of users' short-term intentions and long-term preferences. Subsequently, IPSRec engages a locally reinforced attention structure to learn the migration process of users' short-term intentions and long-term preferences in sequences. Lastly, we combine the user intentions and preferences for the final recommendation. Experiments conducted on various datasets present the preeminence of our approach over several state-of-the-art methods. Moreover, the results corroborate the validity of the feature modeling for user intentions and preferences in accurately representing users' purchase behaviors.

1. Introduction

Traditional recommender systems, such as collaborative filtering (Koren, Bell, & Volinsky, 2009; Sarwar, Karypis, Konstan, & Riedl, 2001; Shen et al., 2021), typically model user–item interactions in a static way, aiming to capture users' general preferences. However, in real recommendation scenarios (Li, Deng, et al., 2024; Liu et al., 2022), user intentions and preferences are dynamic and change over time and context. Thus, sequential recommendation (Wu, He, Wu, Zhang, & Ye, 2023; Wu, Zhong, Yao, & Ye, 2022) has been proposed to model the dynamic sequential relationship over user–item interactions, allowing recommending items accurately in a changing environment.

Extensive researches in sequential recommendation (He & McAuley, 2016a; Hidasi, Karatzoglou, Baltrunas, & Tikk, 2015; Quadrana, Karatzoglou, Hidasi, & Cremonesi, 2017; Rendle, Freudenthaler, & Schmidt-Thieme, 2010; Tang & Wang, 2018; Tuan & Phuong, 2017; Wang et al., 2019; Wu, Ahmed, Beutel, Smola, & Jing, 2017; You et al., 2019) have studied the sequential

* Corresponding authors.
E-mail addresses: bingqian_liu@whu.edu.cn (B. Liu), dtclee1222@whu.edu.cn (D. Li), jianwang@whu.edu.cn (J. Wang), zhihao_wang@whu.edu.cn (Z. Wang), bingli@whu.edu.cn (B. Li).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.ipm.2024.103680
Received 19 September 2023; Received in revised form 4 December 2023; Accepted 28 January 2024

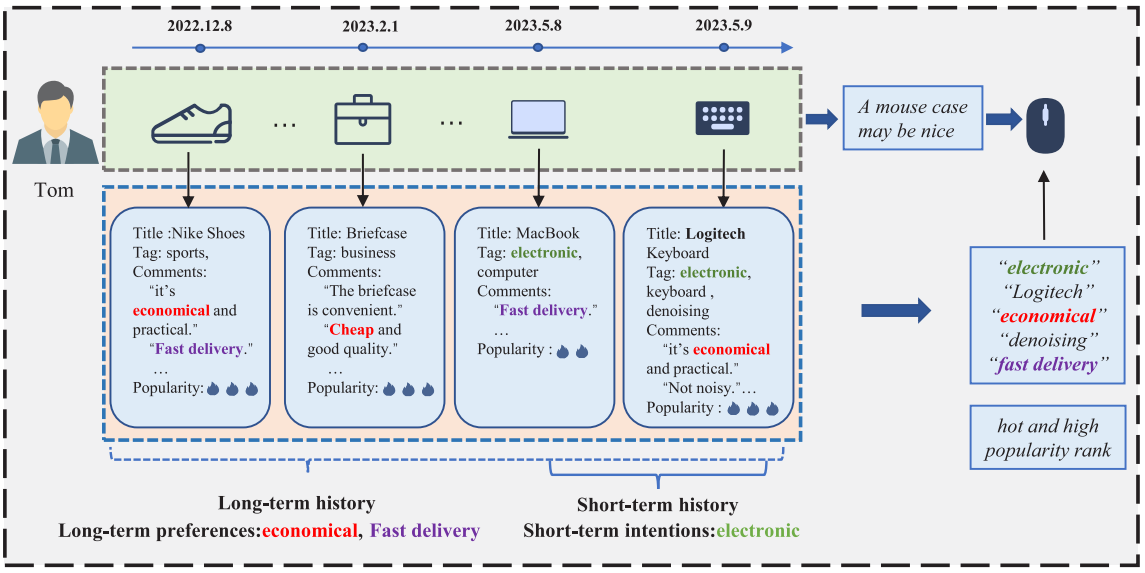

Fig. 1. An example of a user's buying sequence.

relationship between user–item interactions. Markov chain-based approaches (He & McAuley, 2016a; Rendle et al., 2010) construct item transition matrices from historical data to predict user behaviors, but they are limited in capturing long-term dependencies. As deep neural networks evolve (Wang et al., 2023), RNN-based recommendation methods (Hidasi et al., 2015; Quadrana et al., 2017; Wu et al., 2017) and CNN-based recommendation methods (Tang & Wang, 2018; Tuan & Phuong, 2017; You et al., 2019) can represent interactions between users and items in sequential sequences. Considering the nature of dynamic user preferences, the Transformer-based sequential recommendation approaches (Vaswani et al., 2017) model preferences migration based on users' history of interactions (Kang & McAuley, 2018; Sun et al., 2019), and some approaches further improve the attention mechanism to achieve superior performance (He et al., 2021). Despite there have been some works attempting to improve Transformer-based sequence recommendation using relevant auxiliary information (Liu et al., 2021; Rashed, Elsayed, & Schmidt-Thieme, 2022; Xie, Zhou, & Kim, 2022; Zhang et al., 2019), they are limited in simply treating such auxiliary information as feature enhancement of users or items. These methods concatenate auxiliary information with item features or use auxiliary information to calculate the item's weight. Such treatments ignore topological relationships between this auxiliary information and the items, from which they can enrich features by capturing higher-order dependency information, affecting the effectiveness of modeling the item and the auxiliary information. In addition, these methods do not notice that the auxiliary information features and item features may have different degrees of bias in the transformer, causing the degradation of the modeling sequence.

Given the topological correlations among items attribute information and interaction information, graphs provide a more effective structure for characterizing the information (Li, Xia, et al., 2024). Several recent studies have used graph neural networks (Li, Zhang, et al., 2023; Wu, Li, Lin, & Zhang, 2021) for recommender systems (Chang et al., 2021; Fan et al., 2019; Li, Wang, et al., 2021; Shi, Hu, Zhao, & Philip, 2018; Wang, Ding, Hong, Liu, & Caverlee, 2020; Wang, Ding, Zhu, & Caverlee, 2021; Xia et al., 2021; Zhang, Wu, Yu, Liu, & Wang, 2022; Zhang, Xu, et al., 2022). These recommendation approaches map users, items, and attributes onto graph structures to establish topological associations, where various types of information can be explicitly or implicitly connected. Hypergraph-based recommendation methods (Wang et al., 2020, 2021; Xia et al., 2021; Zhang, Wu, et al., 2022; Zhang, Xu, et al., 2022) are able to connect two or more nodes to the same hyperedge, thereby capturing higher-order information between nodes. Heterogeneous graphs (Fan et al., 2019; Li, Wang, Du, Hu, & Yang, 2023; Shi et al., 2018), on the other hand, can handle different types of nodes due to their heterogeneity. These graphs propagate and aggregate information from heterogeneous nodes of different paths. So they can extract semantic information from different perspectives.

In fact, the user's decision is jointly determined by user intentions and user preferences. Fig. 1 illustrates how user intentions and preferences play a significant role in modeling user behaviors. Tom has purchased products like "Nike shoes", "Briefcase", "MacBook", and a "Logitech Keyboard". The titles, categories, and user comments of these products reveal features such as "electronic", "denoising", "fast delivery" and "affordable". These features in purchase history over a longer period can stand for the intrinsic user properties at the attribute level, which is called **long-term user's preferences**. It can also be seen that the user tends to choose items with high popularity, indicating preferences for popularity hierarchy. Meanwhile, after purchasing a "computer" and a "keyboard" in the recent past, Tom may have wanted to purchase an electronic product like a "mouse", which is called **short-term user's intentions**. It represents a user's short-term shopping intentions influenced by recent behaviors. Given these short-term intentions and long-term preferences, recommending a Logitech mouse is more likely to be accepted by the user. Although sequential methods like Transformer-based sequence recommendation utilize the auxiliary information, they disregard the

topological relationships among them and the joint influence of short-term intentions and long-term preferences, which results in degradation of model accuracy. In terms of user behaviors, the users' historical interactions contain heterogeneous information that represents their short-term intentions and long-term preferences. When these two factors are taken into account, users make different item choices. Distinct from previous work (Liu, Li, Wang, Li, & Hang, 2024), our method introduces diverse auxiliary information, constructs heterogeneous hypergraphs to model their relationships, and differentiates learning users' short-term intentions and long-term preferences.

Therefore, we introduce a sequential recommendation method called IPSRec, explicitly mining users' short-term intentions and long-term preferences by fusing heterogeneous auxiliary information from multiple sources. We define users' demand for specific items as their purchase intentions and their favor for items' attributes as their preferences, which is inspired by the Marshall theory of demand in economics as well as the Hayek theory of price. To model user intentions, we introduce item-level intentions representation. Additionally, to fully characterize user preferences, we utilize attribute representations at the conceptual phrase level and item popularity representations at the popularity level. To achieve this, IPSRec constructs a heterogeneous hypergraph module, considering potential topological links among different types of auxiliary information. This allows effective propagation and feature aggregation of diverse auxiliary information, facilitating the interaction among the various data sources. Subsequently, we apply the attention structure to study the migration of users' short-term intentions and long-term preferences in sequences from multiple perspectives. By performing a fusion of different levels of user intentions and preferences, we generate ranking scores for candidate items recommended to users. Experiment results demonstrate that IPSRec exceeds several state-of-the-art sequential recommendation approaches on various datasets.

1.1. Research objective and contributions

The contributions of this paper are recapped below:

- We introduce IPSRec, a sequence recommendation approach that fuses heterogeneous auxiliary information from multiple sources to represent users' short-term intentions and long-term preferences. The proposed approach facilitates the learning of the distinct migration process of users' short-term intentions and long-term preferences through sequences by applying a locally reinforced attention structure.
- To analyze the synergy of user intentions and preferences, we construct a heterogeneous hypergraph network by fusing heterogeneous information from multiple sources. This network can effectively fuse and converge topological information about user intentions and user preferences. As a consequence, it engenders a more appropriate representation of user intentions and preferences to determine the final user behavior.
- We conduct comprehensive experiments on various datasets to compare the IPSRec model with state-of-the-art methods. Experiment results indicate the effectiveness of the method. Moreover, the results also show that the joint modeling of user intentions and preferences can accurately represent user behaviors.

The remaining sections are organized as below: Section 2 investigates related work in sequence recommendation and recommendations based on hypergraphs and heterogeneous graphs. Section 3 introduces the details of the proposed method IPSRec. Next, Section 4 conducts the overall performance comparison, ablation study, hyperparameters analysis, complexity analysis, and visualizing experiment. Lastly, Section 5 and Section 6 summarize the discussion and conclusion.

2. Related work

In this section, we review related work on sequence recommendation and recommendations based on hypergraphs and heterogeneous graphs.

2.1. Sequential recommendation

Early works such as Factorizing Personalized Markov Chains (FPMC) (He & McAuley, 2016a; Rendle et al., 2010) combined matrix decomposition and first-order or higher-order Markov chains to construct user–item transfer matrices to predict user behavior. However, these recommendation results rely only on the last one or a few interactions, ignoring long-term interaction data. With the success of deep neural networks, sequential recommendations based on deep learning are being widely deployed in real-world scenarios. GRU4Rec (Hidasi et al., 2015) introduces RNN for session recommendation, considering the temporal information of the complete sequence. Devooght and Bersini (2017) modifies the RNN to improve the learning of long-term items. However, the RNN-based approach is highly dependent on time step and adjacency items, which cannot model the complex relationships in real recommendation scenarios. In Tang and Wang (2018), the authors propose a CNN-based sequence recommendation algorithm, which can use the convolutional properties of CNNs locally in the sequence, but is limited by the size of the filter that cannot capture long-term dependencies. The attention-based approaches have excellent performance with the attention mechanism and Transformer. SASRec (Kang & McAuley, 2018) learns user interaction sequences unidirectionally according to the structure of the Transformer. BERT4Rec (Sun et al., 2019) introduces the BERT model (Kenton & Toutanova, 2019) to sequence recommendation, modeling user interaction sequences bi-directionally and predicting masked items. HAFLS (Du, Peng, Niu, & Yan, 2022), LSSA (Xu et al., 2021) and CLSR (Zheng et al., 2022) factor in the short- and long-term preferences of users, but the definition of short- and long-term relies on manual segmentation of the entire dataset, which does not allow simulating preferences for a specific individual

user in a well-defined way. These methods learn the dependencies between items in a sequence utilizing an attention mechanism, but only the item IDs are considered without utilizing other available auxiliary information. FDSA (Zhang et al., 2019) applies deep self-attentive networks for patterning the transitions between item features from the feature level. DIF-SR (Xie et al., 2022) inputs auxiliary information to the attention layer and then obtains item representations containing the auxiliary information. While these approaches enrich the representation by using auxiliary information to learn features of items, they ignore the fact that user behavior is jointly determined by users' intentions and preferences, which is crucial for the construction of user behaviors. Although KA-MemNN (Zhu et al., 2020), HIP-RHINE (Yang, Li, & Yue, 2022) and TLSAN (Zhang, Wang, & Yu, 2021) mention the joint role of intentions and preferences, they just utilize the category and interaction information, which cannot fully represent the users' intentions and preferences. To generate a more accurate representation, we further utilize the comments and titles of items besides categories and explicitly differentiate learning about the migration of users' short-term intentions and long-term preferences.

2.2. Recommendations based on hypergraphs and heterogeneous graphs

Hypergraphs and heterogeneous graphs-based approaches are effective in dealing with user–item interactions with complex auxiliary information compared to recommendation methods considering only user–item interactions. MEIRec (Fan et al., 2019) learns node vector representations of heterogeneous graphs based on a meta-path approach, fusing the heterogeneous node representations to predict items that represent user intentions. HERec (Shi et al., 2018) uses a random wandering strategy on meta-paths to generate node sequences, learns node features based on different meta-paths, and merges multiple features to obtain the final node representation. Hyperedges in hypergraphs (Feng, You, Zhang, Ji, & Gao, 2019) can connect two or more vertices to capture higher-order dependencies between nodes. SHARE (Wang et al., 2021) uses a session-based hypergraph attention network to infer user intentions by capturing higher-order relationships of nodes through sliding windows in the context. DHCN (Xia et al., 2021) proposes a two-channel hypergraph convolutional model that conducts graph convolution on hypergraphs and line graphs. H^2SeqRec (Li, Chen, et al., 2021) considers temporal and social information and constructs hypergraphs to learn dynamic item embeddings. GES (Zhu, Sun, & Chen, 2023) considers semantic associations between items and mixes both sequential and semantic relations before learning them through graph convolutional networks. DH-HGCN (Han, Tao, Tang, & Xia, 2022) uses hypergraph convolutional networks to model the dual homogeneity of social relations and item connectivity in order to obtain high-order correlations. DGSR (Zhang, Wu, et al., 2022) and HyperRec (Wang et al., 2020) model dynamic user interaction sequences based on hypergraphs. HCCF (Xia et al., 2022) builds self-supervised tasks that simultaneously capture global and local synergistic relationships through cross-view contrastive learning based on the hypergraphs. HIDE (Li, Gao, Luo, Jin, & Li, 2022) constructs a hypergraph for each user session and models possible interest shifts from different perspectives. These methods do not utilize the diverse auxiliary information of the items themselves and do not make an explicit distinction between learning about user's long-term preferences and short-term intentions. Given the heterogeneity of user intentions and preferences, it is challenging to effectively model their migration process using a simple graph network. Therefore, we introduce a heterogeneous hypergraph network that incorporates the user's multi-source interaction information to model the dynamic changes of user intentions and preferences, as well as their potential synergies.

3. Method

3.1. Problem formulation

This section formally defines the sequence recommendation problem, incorporating an understanding of intentions and preferences. As the set of users $\mathcal{U} = \{u_1, u_2, \dots, u_N\}$ and the set of items $\mathcal{I} = \{i_1, i_2, \dots, i_M\}$ is given, where N, M indicates the number of users and items, the sequential sequences of users' interaction history consist of the interaction records between these N users and M items. Each item $i \in \mathcal{I}$contains side information, including its ID, category, comments, etc. For each user $u \in \mathcal{U}$, the interaction sequence is ordered by the timestamp and denoted as $\mathcal{I}_u = \{i\_(u,1), i\_(u,2), \dots, i\_(u,t)\}$, where item $i\_(u,k)$ denotes an interaction record with user u at the time k.

Besides the interaction sequences of items, IPSRec considers the available descriptive information of items as side information to further learn user preferences. We extract keywords from item titles, categories, and user reviews, transforming them into concept words C that can indicate the intentions of users. C is defined as $C = \{c_1, c_2, \dots, c_Z\}$, where Z means the number of all concept words. The descriptive information of each item $i \in \mathcal{I}$ is represented as a combination of several concept words $c'_i = (c_a, c_b, \dots)$, where c'_i denotes the combination of concept words, $c_a, c_b \in C$ denote the concept words that the item may contain and the number of concept words contained is between 1 and Z. For each user $u \in \mathcal{U}$, the concept word combinations sequence corresponding to the interaction sequence is $C'_u = \{c'_{i\_(u,1)}, c'_{i\_(u,2)}, \dots, c'_{i\_(u,t)}\}$, where $c'_{i\_(u,k)}$ indicates that user u interacted with item $i\_(u,k)$ at time k, and the combination of concept words of that item is $c'_{i\_(u,k)}$. In consideration of the fact that some users interacted with items far below the average, recommending the next item for users may suffer from a lack of sufficient interaction information to extract user preferences and intentions. In this case, IPSRec prioritizes recommending items that match the users' long-term preferences in the popularity level by considering the popularity ranking of items. For each item $i \in \mathcal{I}$, the popularity ranking p_i of the item is obtained based on its occurrences in all interaction records. $\mathcal{P} = \{p_1, p_2, \dots, p_Q\}$ indicates the popularity rankings of all items where Q is the number of different popularity rankings. For user $u \in \mathcal{U}$, the popularity ranking sequence is $\mathcal{P}_u = \{p_{i\_(u,1)}, p_{i\_(u,2)}, \dots, p_{i\_(u,t)}\}$, where $p_{i\_(u,k)}$ means that user u interacted with item $i\_(u,k)$ at time k, and the item popularity ranking is $p_{i\_(u,k)}$.

With all the above information, the sequential recommendation can be specified formally as predicting the probability of interacting with item $i\_(u,t+1)$ at the moment $t+1$ for each user $u \in \mathcal{U}$:

$$Y(i\_(u,t+1) | \mathcal{I}_u \ , \ \mathcal{P}_u \ , \ C'_u). \tag{1}$$

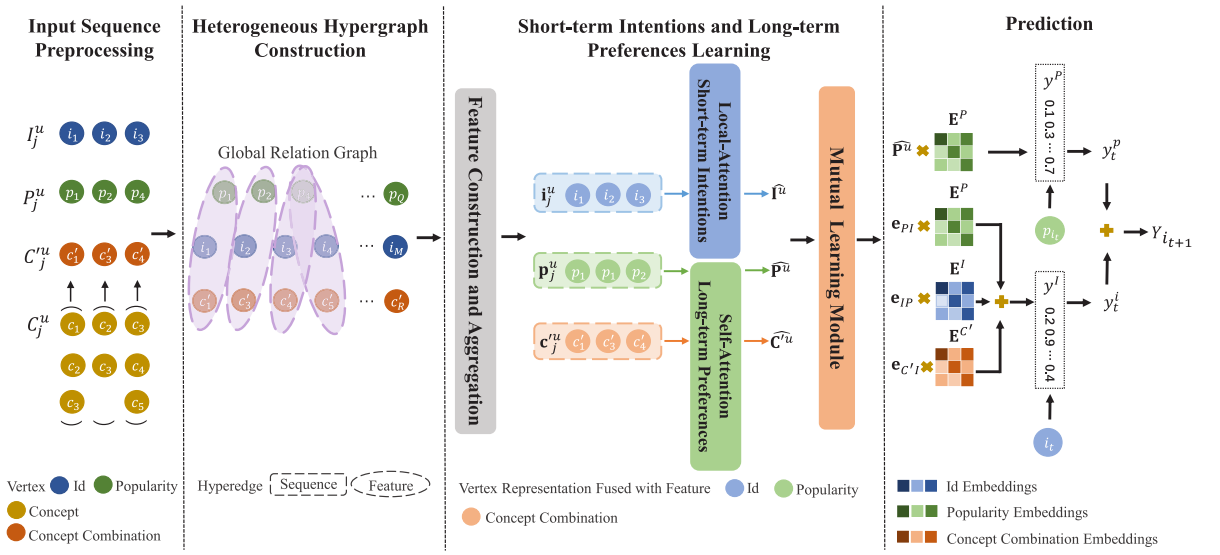

Fig. 2. The overall network architecture of IPSRec.

3.2. Model framework

Fig. 2 shows the framework of IPSRec. IPSRec contains four modules: input sequence preprocessing, heterogeneous hypergraph construction, short-term intentions and long-term preferences learning, and model prediction. Firstly, IPSRec preprocesses items and their side information and converts the original information into dense vectors. To represent global relationships, we design a heterogeneous hypergraph built on user interaction sequences. The heterogeneous information is propagated and aggregated through the hyperedges. Then, IPSRec performs self-attention and locally constrained self-attention to grasp users' long-term preferences and users' short-term intentions. Taking the influence of different user intentions and preferences on each other into account, the mutual learning module is applied to fuse the various intentions and preferences. Finally, IPSRec predicts the next item according to the user's intentions and preferences. We will subsequently provide a detailed description of these modules. As for the notations we defined, they are summarized in Table 1.

3.3. Input sequence preprocessing

For all items $\mathcal{I}$ and items' popularity rankings $\mathcal{P}$, IPSRec applies an embedding layer for feature representation. When processing concept words that represent the characteristics of items, it can be observed that the descriptions of items contain some conceptual information. This module processes the keywords extracted from items' descriptions into concept words inspired by ISRec (Li, Wang, et al., 2021). The specific steps are as follows: (1) Extracting: IPSRec uses the RAKE algorithm (Rose, Engel, Cramer, & Cowley, 2010) to calculate the frequency of word occurrences and analyze their co-occurrence with other words in the text, and then extracts keywords from the available descriptive information such as titles, comments, and categories of items. (2) Filtering: Having obtained the extracted keywords, IPSRec filters out words used less than 0.5%. Additionally, the words with no practical meanings, such as pronouns, adverbs, and prepositions, are removed manually. The words, like nouns, adjectives, and adverbs, which can represent the characteristics of the items, are left and recorded as the set of concept words $\mathcal{C}$. (3) Combining: In order to represent multiple semantic meanings of different concept words, IPSRec defines the concept word combination c'_i for item $i \in \mathcal{I}$. Each item perhaps contains some different concept words. The number of concept words ranges from 1 to Z, where Z indicates the number of all concept words. Different combinations of concept words show different meanings. An item corresponds to a specific combination of concept words. The set of concept word combinations for all items is $\mathcal{C}' = \{c'_1, c'_2, \ldots, c'_R\}$ where R is the number of all concept word combinations. The initial embedding matrices of items, item popularity rankings, and item concept word combinations can be obtained as follows:

$$\mathbf{E}^I = Embedding(\{i_1, i_2, \ldots, i_M\}), \tag{2}$$

$$\mathbf{E}^P = Embedding(\{p_1, p_2, \ldots, p_Q\}), \tag{3}$$

$$\mathbf{E}^{C'} = Embedding(\{c'_1, c'_2, \ldots, c'_R\}), \tag{4}$$

where $\mathbf{E}^I \in \mathbb{R}^{M \times d}$, $\mathbf{E}^P \in \mathbb{R}^{Q \times d}$, $\mathbf{E}^{C'} \in \mathbb{R}^{R \times d}$ and d is the vector dimension.

Table 1
Notations and description.

Notations	Description
$\mathcal{U}$, $\mathcal{I}$, $\mathcal{C}$, $\mathcal{C}'$, $\mathcal{P}$	user set, item set, concept words set, concept word combinations set, popularity ranking set
N, M, Z, R, Q	number of users, number of items, number of concept words, number of concept word combinations, number of popularity rankings
$\mathcal{I}_u$, $\mathcal{C}'_u$, $\mathcal{P}_u$	*user–item* sequence, *user-concept word combinations* sequence, *user-popularity ranking* sequence
t, k	index of the time
T	maximum sequence length
Y	the probability that users interact with items
d	vector dimensionality
$\mathbf{E}^I \in \mathbb{R}^{M\times d}$, $\mathbf{E}^P \in \mathbb{R}^{Q\times d}$, $\mathbf{E}^{C'} \in \mathbb{R}^{R\times d}$	item embedding matrix, popularity ranking embedding matrix, concept word combinations embedding matrix
$\mathbf{e}_i \in \mathbb{R}^d$, $\mathbf{e}_p \in \mathbb{R}^d$, $\mathbf{e}_{c'} \in \mathbb{R}^d$	embedding vector of item i, embedding vector of popularity ranking p, embedding vector of concept word combinations c'
$\widehat{\mathbf{I}}^u, \widehat{\mathbf{P}}^u, \widehat{\mathbf{C}'}^u$	embedding matrix of *user–item* sequence, *user-concept word combinations* sequence, *user-popularity ranking* sequence after attention mechanism learning
$\mathcal{G}$	heterogeneous hypergraph
$\mathcal{V}$	vertex set of the graph
$\mathcal{A}$	adjacency matrix set
$\mathbf{A}_{V^{(1)}V^{(2)}} \in \mathbb{R}^{\lvert V^{(1)}\rvert\times\lvert V^{(2)}\rvert}$	adjacency matrix consisting of nodes of type $\mathcal{V}^{(1)}$ and nodes of type $\mathcal{V}^{(2)}$
$\mathcal{E}, \mathcal{E}_s, \mathcal{E}_f$	hyperedge set, sequence hyperedge set, feature hyperedge set
ε_s^θ	a sequence hyperedge of type θ
ε_f^i	feature hyperedge of item i
$\mathbf{POS} \in \mathbb{R}^{T\times d}$	position embedding matrix
y_t^I, y_t^P	recommended item score, recommended popularity ranking score
$\mathcal{L}$	loss function

3.4. Heterogeneous hypergraph construction

To represent the high-order dependencies among items and items, information and information, and items and information, IPSRec constructs a heterogeneous hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. $\mathcal{V}$ includes the set of items $\mathcal{I}$, the set of concept word combinations $\mathcal{C}'$ and the set of popularity rankings $\mathcal{P}$, i.e., $\mathcal{V} = \mathcal{I} \cup \mathcal{C}' \cup \mathcal{P}$. $\mathcal{E}$ is the set of hyperedges. There are two types of hyperedges, sequence hyperedges $\mathcal{E}_s$ and feature hyperedges $\mathcal{E}_f$, i.e., $\mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_f$. For each sequence hyperedge $\varepsilon_s^\theta \in \mathcal{E}_s$, ε_s^θ connects nodes located at this sequence, where θ denotes one of the three types, i.e., ε_s^θ, $\theta \in \mathcal{I} \cup \mathcal{C}' \cup \mathcal{P}$. For each feature hyperedge $\varepsilon_f^i \in \mathcal{E}_f$, ε_f^i connects heterogeneous information for an item $i \in \mathcal{I}$, i.e., $\varepsilon_f^i =< i, p_i, c'_i >$. $\mathcal{A}$ is the set of adjacency matrices which stores the heterogeneous relationships of any two of the three types of nodes $\mathcal{I}$, $\mathcal{C}'$, $\mathcal{P}$, i.e., $\mathcal{A} = \mathbf{A}_{IP} \cup \mathbf{A}_{IC'} \cup \mathbf{A}_{PI} \cup \mathbf{A}_{PC'} \cup \mathbf{A}_{C'I} \cup \mathbf{A}_{C'P}$. If the two heterogeneous nodes are connected based on the feature hyperedge, then the corresponding value in the adjacency matrix is set to 1, indicating the existence of an adjacency relationship between these two nodes, and it is set to 0 otherwise. For two arbitrary heterogeneous nodes $\mathcal{V}^{(1)}$, $\mathcal{V}^{(2)}$:

$$\mathbf{A}_{V^{(1)}V^{(2)}}[i][j] = \begin{cases} 1, & \text{if } i \in \mathcal{E}_f \text{ and } j \in \mathcal{E}_f \\ 0, & \text{else}, \end{cases} \tag{5}$$

where $\mathbf{A}_{V^{(1)}V^{(2)}} \in \mathbb{R}^{\left|V^{(1)}\right|\times\left|V^{(2)}\right|}$, $i \in \mathcal{V}^{(1)}$ and $j \in \mathcal{V}^{(2)}$. $\mathcal{V}^{(1)}$ and $\mathcal{V}^{(2)}$ are of different types.

IPSRec learns the sequential information to extract the user's intentions and preferences from the sequence hyperedge and propagates the information of heterogeneous nodes to aggregate features based on feature hyperedge. If nodes are connected through a hyperedge, they are adjacent.

3.5. Message propagation

A node can be a neighbor to many nodes of different types. Nodes on feature hyperedges contain heterogeneous information. For the processing of high-order complex heterogeneous information, IPSRec applies both intra-aggregation and inter-aggregation to transfer information between nodes based on feature hyperedges.

3.5.1. Intra aggregation

(1) Intra Aggregation for Items Nodes

As the initial embedding matrix of the items $\mathbf{E}^I$ has been generated, for each item node $v_i \in \mathcal{I}$, its feature is represented as the ith vector $\mathbf{e}_i \in R^d$ in the matrix. The popularity ranking node's feature $\mathbf{e}_p$ and the concept word combination node's feature $\mathbf{e}_{c'}$ are aggregated to the corresponding item node, where these nodes are adjacent on the feature hyperedges. The representation

of the item node with popularity ranking feature $\mathbf{e}_i^p$ and the representation $\mathbf{e}_i^{c'}$ with the concept word combination feature can be drawn from the following:

$$\mathbf{e}_i^p = \mathbf{A}_{IP}[i][p]\mathbf{e}_p, \mathbf{e}_i^{c'} = \mathbf{A}_{IC'}[i][c']\mathbf{e}_{c'}, \tag{6}$$

where $\mathbf{e}_p, \mathbf{e}_{c'}, \mathbf{e}_i^p, \mathbf{e}_i^{c'} \in \mathbb{R}^d$.

(2) Intra Aggregation for Side Information Nodes

IPSRec employs the attention-based method to perform intra-aggregation for popularity ranking nodes and concept word combination nodes. The attention weights distinguish the importance of nodes with different types by assigning distinct scores to those nodes. The embedding vector of the side information node v_s of type S is $\mathbf{e}_s \in \mathbb{R}^d$ and the set of its neighbors with type η is denoted as N_s^η, where $S \in \mathcal{P} \cup C'$, $\eta \in (\mathcal{V} - S)$. For each neighbor node $v_i^\eta \in N_s^\eta$ and its node feature $\mathbf{e}_i^\eta \in \mathbb{R}^d$, its attention score is calculated as follows:

$$p_{i\eta} = \boldsymbol{\rho}_\eta \mathbf{e}_i^\eta, \tag{7}$$

where ${\boldsymbol{\rho}_\eta}^T \in \mathbb{R}^d$ is the learnable parameter vector, and different attention vectors are applied to extract features for side information nodes with various types. $p_{i\eta}$ denotes the importance of neighbor node v_i^η with type η on target node v_s. $p_{i\eta}$ are utilized for calculating the attention weights for linear calculations:

$$\alpha_{i\eta} = \frac{\exp(p_{i\eta})}{\sum_{v_i^\eta \in N_s^\eta} \exp(p_{i\eta})}. \tag{8}$$

The feature expression of node v_s on type η is:

$$\mathbf{e}_s^\eta = \sum_{v_i^\eta \in N_s^\eta} \alpha_{i\eta} \mathbf{e}_i^\eta, \tag{9}$$

where $\mathbf{e}_s^\eta \in \mathbb{R}^d$.

3.5.2. Inter aggregation

After the node representations $\mathbf{e}_i^p$, $\mathbf{e}_i^{c'}$, $\mathbf{e}_s^\eta$ are obtained by intra aggregation, inter aggregation gathers heterogeneous information from different types of nodes to target nodes. Feature aggregation of item nodes is illustrated as an example:

$$f_i = Sigmoid(\mathbf{W}_i^p \mathbf{e}_i^p + \mathbf{W}_i^{c'} \mathbf{e}_i^{c'} + \mathbf{W}_i[\mathbf{e}_i, \mathbf{e}_i^p, \mathbf{e}_i^{c'}]), \tag{10}$$

where $[,]$ denotes the concatenation operation, $\mathbf{W}_i \in \mathbb{R}^{d\times 3d}$, $\mathbf{W}_i^p$, $\mathbf{W}_i^{c'} \in \mathbb{R}^{d\times d}$ are learnable parameters, and f_i shows correlations between the original node representations and the node representations after fusing heterogeneous side information. In order to make the fusion of features without destroying the original information of the target node, IPSRec further specifies:

$$\mathbf{e}_i^I = \mathbf{e}_i + f_i \cdot \mathbf{e}_i^p + (1 - f_i) \cdot \mathbf{e}_i^{c'}, \tag{11}$$

where $\cdot$ denotes the dot product, $\mathbf{e}_i^I \in \mathbb{R}^d$ represents the ith item node representation after feature fusion and semantic enrichment by heterogeneous nodes. For these three types of nodes, IPSRec repeats the above steps to get the representation of each target node and updates the embedding matrices to obtain the final embedding matrices $\mathbf{E}^I \in \mathbb{R}^{M\times d}$, $\mathbf{E}^P \in \mathbb{R}^{Q\times d}$, $\mathbf{E}^{C'} \in \mathbb{R}^{R\times d}$.

3.6. Short-term intentions and long-term preferences learning

Once nodes' representations fused with heterogeneous information are obtained, IPSRec performs the users' short-term intentions and long-term preferences learning through sequence hyperedge. Since sequences contain sequential information, IPSRec uses the attention structure to model sequence features. Compared with the popularity ranking sequences and concept word combination sequences, the item sequences are sparser and contain more specific user intentions. The influence of short-term intentions is more significant. IPSRec applies a locally constrained self-attention mechanism for item sequences to increase the ability to focus on short-term items and learn short-term user intentions. IPSRec applies a multi-head self-attention mechanism for popularity ranking sequences and concept word combination sequences to learn long-time features of the sequences and extract user preferences.

The training sequences $\mathcal{I}^u$, $\mathcal{P}^u$, $\mathcal{C}\prime^u$ are processed as sequences with fixed length T, where T denotes the maximum length of the sequences. If the length exceeds T, the most recent T interactions are considered. If the length is less than T, the padding item is added repeatedly on the left side until the length equals T.

3.6.1. Self-attention mechanism for long-term preferences learning

The multi-head self-attention structure is applied to popularity ranking sequences and concept word combination sequences to extract users' long-term preferences for different popularity levels and attributes. To represent the positions of the items, the learnable positional embedding $\mathbf{POS} = [\mathbf{pos}_1, \mathbf{pos}_2, \ldots, \mathbf{pos}_T] \in \mathbb{R}^{T\times d}$ is added, where $\mathbf{pos}_i$ indicates the positional embedding of position i. Each item in the sequence of type θ is represented as follows:

$$\mathbf{h}_i^\theta = \mathbf{e}_i^\theta + \mathbf{pos}_i, \tag{12}$$

$\mathbf{h}_i^\theta$ is used as the initial input and $\mathbf{H}_1^\theta$ is used as the input to the first multi-head self-attention layer:

$$\mathbf{H}_1^\theta = [\mathbf{h}_{1\_(1)}^\theta, \mathbf{h}_{2\_(1)}^\theta, \dots, \mathbf{h}_{T\_(1)}^\theta]. \tag{13}$$

The scaled dot product attention is calculated as below:

$$Attention\left(\mathbf{Q}_i, \mathbf{K}_j, \mathbf{V}_j\right) = Softmax(\frac{\mathbf{Q}_i \mathbf{K}_j^T}{\sqrt{d}})\mathbf{V}_j, \tag{14}$$

where $\mathbf{Q}_i$ represents the ith query, $\mathbf{K}_j$ represents the jth key, and $\mathbf{V}_j$ represents the jth value. Due to the temporal nature of the sequence, for the ith item, only the items before the ith item in the sequence are visible and available, so we set $i \geq j$. d is the vector dimension. $\sqrt{d}$ is the scaling factor to avoid large inner product values. Here, the query, key, and value are calculated from the item representation. In the lth multi-head self-attention layer, the sequence representation $\mathbf{H}_l^\theta$ is projected through the parameter matrix and then used as input to calculate attention scores within heads. The scores of each head are concatenated to obtain the output of the lth self-attention layer:

$$\mathbf{S}_{l(i)}^\theta = Attention(\mathbf{H}_l^\theta \mathbf{W}_{Q(l)(i)}^\theta, \mathbf{H}_l^\theta \mathbf{W}_{K(l)(i)}^\theta, \mathbf{H}_l^\theta \mathbf{W}_{V(l)(i)}^\theta), \tag{15}$$

$$\mathbf{S}_l^\theta = Concat[\mathbf{S}_{l(1)}^\theta, \mathbf{S}_{l(2)}^\theta, \dots, \mathbf{S}_{l(m)}^\theta]\mathbf{W}_l^\theta, \tag{16}$$

where $\mathbf{W}_{Q(l)(i)}^\theta$, $\mathbf{W}_{K(l)(i)}^\theta$, $\mathbf{W}_{V(l)(i)}^\theta \in \mathbb{R}^{d \times \frac{d}{m}}$ are the parameters of the ith head of the lth multi-head self-attention layer, m denotes the total number of heads in the self-attention layer, and $\mathbf{W}_l^\theta \in \mathbb{R}^{d \times d}$ are the parameters of the lth multi-head self-attention layer for mapping the results of concatenation to the output. In an effort to make the model nonlinear, prevent results from overfitting, and train more consistently, layer normalization, dropout, residual connection, and position-wise feed-forward network are applied at each layer. The output of the layer is as follows:

$$\mathbf{H}_{l+1}^\theta = FFN\left(\mathbf{S}_l^\theta\right) = ReLU\left(\mathbf{S}_l^\theta \mathbf{W}_{1(l)}^\theta + b_{1(l)}^\theta\right)\mathbf{W}_{2(l)}^\theta + b_{2(l)}^\theta, \tag{17}$$

where $\mathbf{W}_{1(l)}^\theta$, $\mathbf{W}_{2(l)}^\theta \in \mathbb{R}^{d \times d}$ and $b_{1(l)}^\theta$, $b_{2(l)}^\theta \in \mathbb{R}^d$ are the parameters of the lth feed-forward network. Through such L layers, each item in the sequence incorporates the information of the items located before it. The final sequence representation $\mathbf{H}_L^\theta$ is obtained, i.e., the user's preferences for popularity rankings and preferences for concept word combinations:

$$\widehat{\mathbf{P}^u} = \mathbf{H}_L^P, \tag{18}$$

$$\widehat{\mathbf{C}'^u} = \mathbf{H}_L^{C'}. \tag{19}$$

3.6.2. Local-attention mechanism for short-term intentions learning

The locally constrained multi-head self-attention mechanism is used for short-term intentions learning. For reinforcing the ability to focus on short-term items, the attention weight calculation in module 3.6.1 is modified. Inspired by the Locker (He et al., 2021), the distance matrix $\mathbf{D} \in \mathbb{R}^{2T \times \frac{d}{m}}$ between items is generated, where T is the maximum length of the sequence, d is the dimension of the vector and m denotes the total number of heads in the attention layer. To prevent negative distance, we calculate the distance between items as $d_{ij} = T + i - j$. Items information and *item–item* distances are introduced to improve the location-aware attention weight calculation:

$$AttentionOf\,Locker\left(\mathbf{Q}_i, \mathbf{K}_j, \mathbf{V}_j\right) = Softmax(\frac{\mathbf{Q}_i \mathbf{K}_j^T}{\sqrt{d}} + \mu_{ij})\mathbf{V}_j, \tag{20}$$

where μ_{ij} is the factor about the influence of item information and distances.

$$\mathbf{n}_{ij} = \mathbf{Q}_i + \mathbf{K}_j + \mathbf{distance}_{ij}, \tag{21}$$

$$\mu_{ij} = \mathbf{n}_{ij}\mathbf{W}_{l(m)} + b_{l(m)}, \tag{22}$$

where $\mathbf{distance}_{ij} \in \mathbb{R}^{\frac{d}{m}}$ denotes the d_{ij}th vector in the distance matrix $\mathbf{D}$ and $\mathbf{W}_{l(m)} \in \mathbb{R}^{\frac{d}{m} \times 1}$, $b_{l(m)} \in \mathbb{R}^1$ denote the parameters of the mth head of the lth locally constrained self-attention layer. While learning short-term items, long-term historical information also has an impact on the results. Therefore, IPSRec uses the general self-attention mechanism for g heads (same as the calculation of attention weights in module 3.6.1), and the locally constrained self-attention mechanism for the remaining $(m - g)$ heads. The outcomes of all heads are concatenated:

$$\mathbf{S}_{l(i)}^I = Attention\left(\mathbf{H}_l^I \mathbf{W}_{Q(l)(i)}^I, \mathbf{H}_l^I \mathbf{W}_{K(l)(i)}^I, \mathbf{H}_l^I \mathbf{W}_{V(l)(i)}^I\right), \tag{23}$$

$$\mathbf{SL}_{l(j)}^I = AttentionOf\,Locker\left(\mathbf{H}_l^I \mathbf{O}_{Q(l)(j)}^I, \mathbf{H}_l^I \mathbf{O}_{K(l)(j)}^I, \mathbf{H}_l^I \mathbf{O}_{V(l)(j)}^I\right), \tag{24}$$

$$\mathbf{S}_l^I = Concat[\mathbf{S}_{l(1)}^I, \dots, \mathbf{S}_{l(g)}^I, \mathbf{SL}_{l(g+1)}^I, \dots \mathbf{SL}_{l(m)}^I]\mathbf{W}_l^I, \tag{25}$$

where $\mathbf{W}^I_{Q(l)(i)}$, $\mathbf{W}^I_{K(l)(i)}$, $\mathbf{W}^I_{V(l)(i)} \in \mathbb{R}^{d\times\frac{d}{m}}$ are the parameters of the ith general attention head, $\mathbf{O}^I_{Q(l)(j)}$, $\mathbf{O}^I_{K(l)(j)}$, $\mathbf{O}^I_{V(l)(j)} \in \mathbb{R}^{d\times\frac{d}{m}}$ are the parameters of the jth locally constraint attention head in the lth locally constrained multi-head self-attention layer, and $\mathbf{W}^I_l \in \mathbb{R}^{d\times d}$ are the parameters of the lth locally constrained multi-head self-attention layer. After such L layers, the user's intentions for the item $\widehat{\mathbf{I}}^u$ are extracted from $\mathbf{H}^I_L$.

3.7. Mutual learning schema

The user's short-term intentions for the item $\widehat{\mathbf{I}}_u$, long-term preferences for popularity rankings $\widehat{\mathbf{P}}_u$ and preferences for concept words combinations $\widehat{\mathbf{C}'_u}$ have been obtained. These intentions and preferences should not be considered singularly when making recommendations for the user because these factors influence each other. In the situation that multiple preferences jointly determine the user's choice, simple adding or concatenating operations cannot simulate this complex relationship. IPSRec fuses the semantic meanings of $\widehat{\mathbf{I}}_u$ and $\widehat{\mathbf{P}}_u$, and meanings of $\widehat{\mathbf{I}}_u$ and $\widehat{\mathbf{C}'_u}$ based on the gating mechanism inspired by the CoHHN (Zhang, Xu, et al., 2022). IPSRec merges them in the following two perspectives:

$$\boldsymbol{X}_m = Tanh(\mathbf{W}^{IP}_1\left(\widehat{\mathbf{I}}^u \cdot \widehat{\mathbf{P}}^u\right) + b^m_1), \tag{26}$$

$$\boldsymbol{X}_a = Tanh(\mathbf{W}^{IP}_2\left(\widehat{\mathbf{I}}^u + \widehat{\mathbf{P}}^u\right) + b^a_1), \tag{27}$$

$$\boldsymbol{X}'_m = Tanh(\mathbf{W}^{IC'}_1\left(\widehat{\mathbf{I}}^u \cdot \widehat{\mathbf{C}'^u}\right) + b^m_2), \tag{28}$$

$$\boldsymbol{X}'_a = Tanh(\mathbf{W}^{IC'}_2\left(\widehat{\mathbf{I}}^u + \widehat{\mathbf{C}'^u}\right) + b^a_2), \tag{29}$$

where $\mathbf{W}^{IP}_1$, $\mathbf{W}^{IP}_2$, $\mathbf{W}^{IC'}_1$, $\mathbf{W}^{IC'}_2 \in \mathbb{R}^{d\times d}$ and b^m_1, b^a_1, b^m_2, b^a_2 are learnable parameters. IPSRec treats $\boldsymbol{X}_m$ and $\boldsymbol{X}'_m$ as the "the current input", and treats $\boldsymbol{X}_a$ and $\boldsymbol{X}'_a$ as the "the state at the previous moment" in the gating mechanism. The two meanings are approximately fused based on resetting gate and updating gate. Take $\boldsymbol{X}_m$, $\boldsymbol{X}_a$ for example:

$$\boldsymbol{J}_1 = Sigmoid(\mathbf{W}^{\delta}_1\boldsymbol{X}_m + \mathbf{O}^{\delta}_1\boldsymbol{X}_a), \tag{30}$$

$$\boldsymbol{J}_2 = Sigmoid(\mathbf{W}^{\delta}_2\boldsymbol{X}_m + \mathbf{O}^{\delta}_2\boldsymbol{X}_a), \tag{31}$$

where $\mathbf{W}^{\delta}_1$, $\mathbf{O}^{\delta}_1$, $\mathbf{W}^{\delta}_2$, $\mathbf{O}^{\delta}_2 \in \mathbb{R}^{d\times d}$ are the learnable parameters. $\boldsymbol{J}_1$ is the resetting gate to control the effect of $\boldsymbol{X}_a$ on $\boldsymbol{X}_m$. $\boldsymbol{J}_2$ is the updating gate to decide whether to ignore the effect of $\boldsymbol{X}_m$. After performing the update of the two meanings, we obtain the fused popularity-item and item-popularity representations.

$$\mathbf{M}_I = Tanh(\mathbf{W}^I\left(\boldsymbol{J}_1 \cdot \widehat{\mathbf{I}}^u\right) + \mathbf{O}^I((1-\boldsymbol{J}_2)\cdot\widehat{\mathbf{P}}^u)), \tag{32}$$

$$\mathbf{M}_P = Tanh(\mathbf{W}^P\left(\boldsymbol{J}_2 \cdot \widehat{\mathbf{P}}^u\right) + \mathbf{O}^P((1-\boldsymbol{J}_1)\cdot\widehat{\mathbf{I}}^u)), \tag{33}$$

$$\mathbf{E}_{PI} = \mathbf{M}_I \cdot (\widehat{\mathbf{I}}^u + \mathbf{M}_P), \tag{34}$$

$$\mathbf{E}_{IP} = \mathbf{M}_P \cdot (\widehat{\mathbf{I}}^u + \mathbf{M}_I), \tag{35}$$

where $\mathbf{W}^I$, $\mathbf{O}^I$, $\mathbf{W}^P$, $\mathbf{O}^P \in \mathbb{R}^{d\times d}$ are learnable parameters. $\mathbf{E}_{IP}$ is the user's item intentions fused with the information of preferences for popularity rankings. $\mathbf{E}_{PI}$ is the user's popularity ranking preferences fused with the information of item intentions. The user's preferences for concept word combinations $\mathbf{E}_{C'I}$ can be obtained by performing the above operations on $\boldsymbol{X}'_m$ and $\boldsymbol{X}'_a$.

3.8. Objective function and prediction

Based on the user's item intentions $\mathbf{E}_{IP}$, preferences for popularity rankings $\mathbf{E}_{PI}$, preferences for concept word combinations $\mathbf{E}_{C'I}$ and embedding matrices $\mathbf{E}^I$, $\mathbf{E}^P$, $\mathbf{E}^{C'}$, the final likelihood score y^I_t at timestamp t for all items is calculated as below:

$$\mathbf{E}^P_I = \mathbf{A}_{V^{(I)}V^{(P)}}\mathbf{E}^P, \quad \mathbf{E}^{C'}_I = \mathbf{A}_{V^{(I)}V^{(C')}}\mathbf{E}^{C'}, \tag{36}$$

$$y^I_t = \mathbf{e}_{IP}(\mathbf{E}^I)^T + \mathbf{e}_{PI}(\mathbf{E}^P_I)^T + \mathbf{e}_{C'I}(\mathbf{E}^{C'}_I)^T, \tag{37}$$

where $\mathbf{e}_{IP}$, $\mathbf{e}_{PI}$, $\mathbf{e}_{C'I}$ are the intentions and preferences representations of $\mathbf{E}_{IP}$, $\mathbf{E}_{PI}$, $\mathbf{E}_{C'I}$ at timestamp t.

To make the recommended items more acceptable and accurate to the user, we calculate the likelihood score y^p_t of the popularity ranking recommendation using the user-popularity ranking preferences $\widehat{\mathbf{P}}^u$ that is not fused with the user–item sequence $\widehat{\mathbf{I}}^u$. y^P_t indicates the final likelihood score for all popularity levels at timestamp t:

$$y^P_t = \hat{\mathbf{p}}(\mathbf{E}^P_I)^T, \tag{38}$$

where $\hat{\mathbf{p}}$ is the representation of $\widehat{\mathbf{P}}^u$ at timestamp t.

We train the model following the training methods of prior sequential recommendations. The negative log-likelihood function is applied as the objective function:

$$\mathcal{L} = \mathcal{L}_{I_u} + \mathcal{L}_{P_u} + \alpha \|\omega\|_2^2, \tag{39}$$

$$y_t^i = Prob(i_{t+1}|[i_1, i_2, \dots, i_t]) = Softmax(y_t^I), \tag{40}$$

$$y_t^p = Prob(p_{t+1}|[p_1, p_2, \dots, p_t]) = Softmax(y_t^P), \tag{41}$$

$$\mathcal{L}_{I_u} = \frac{1}{|\mathcal{U}|}\frac{1}{|\mathcal{I}_u|}\sum_{u\in\mathcal{U}}\sum_{t\in\mathcal{I}_u} -\log(y_t^i), \tag{42}$$

$$\mathcal{L}_{P_u} = \frac{1}{|\mathcal{U}|}\frac{1}{|\mathcal{I}_u|}\sum_{u\in\mathcal{U}}\sum_{t\in\mathcal{I}_u} -\log(y_t^p), \tag{43}$$

where y_t^i, y_t^p means the score of the user interacting with the next item and its popularity level at timestamp t, α indicates the regularization factor, and ω represents the model parameter. $\mathcal{L}_{I_u}$ is the loss of the item intentions and $\mathcal{L}_{P_u}$ is the loss of the popularity level preferences.

When predicting the next item, the probability $Y(i\_(u,t+1)$ that the user u interacts with item i at time $t+1$ is equal to $Y_{i_{t+1}} = y_t^i + y_t^p$.

4. Experiments

We conduct experiments to evaluate IPSRec on widely used datasets, set up ablation experiments to examine the efficacy of each component, explore the capability of hyperparameters, analyze the model complexity, and visualize attention weight in this section.

4.1. Datasets

We conduct experiments on the following datasets from real platforms.

- MovieLens (Harper & Konstan, 2015)[2]: This dataset collects movie ratings from the MovieLens website, which is commonly used to evaluate recommendation algorithms. We select MovieLens-1M(ml-1 m) and MovieLens-20M(ml-20 m). MovieLens-1M contains 1 million ratings of about 4000 movies by about 6000 users. As for MovieLens-20M, it contains 20 million ratings of about 27,000 movies by about 13,800 users. We extract user interaction records from the rating records. The keywords are extracted from movie titles and categories and filtered as concept words.
- Amazon (He & McAuley, 2016b; McAuley, Targett, Shi, & Van Den Hengel, 2015)[3]: This dataset is widely used for recommendation research. It contains user reviews and product metadata from Amazon.com. We select one of the datasets with the category "Beauty", which has approximately 2,000,000 reviews and 260,000 products. We extract interaction records from the review data and keywords from product titles and reviews.
- Steam (Kang & McAuley, 2018; Pathak, Gupta, & McAuley, 2017; Wan & McAuley, 2018)[4]: This dataset is a collection of reviews and game metadata from the Steam video game platform, containing 7,790,000 reviews from approximately 2,570,000 users on 15,474 games. We extract user interaction records from review data and keywords from game names and reviews.

Our processing of the datasets follows previous work (Kang & McAuley, 2018; Li, Wang, et al., 2021; Sun et al., 2019; Tang & Wang, 2018), where a user score or comment is considered as an interaction between that user and the item. Users and items whose interaction records are fewer than 5 were removed. We group interaction records by the user and generate interaction sequences based on timestamp or date.

For the calculation of popularity ranking, we count the number of occurrences of each item in each dataset and divide the number of occurrences of each item by the total number of interactions as the value of popularity ranking. We split the values into ten popularity levels to construct popularity feature vectors. We follow the strategy in Li, Wang, et al. (2021) and use the RAKE algorithm (Rose et al., 2010) to extract concept words from the available descriptive information of the dataset. For the MovieLens dataset, we extract keywords from the attributes "title" and "genre" in the item metadata. For the Amazon and Steam datasets, we extract keywords from user comments in the interaction records and from the titles (or names) of the item metadata. After extracting keywords from each dataset, we remove words whose frequency is less than 0.5% in that dataset. In addition, in order to make the extracted keywords represent the actual attributes of the items and the real preferences of users, we manually filtered out pronouns, adverbs, crowns, and words without real meaning, such as "it", "what", "nothing", and so forth, because these words cannot convey effective information. The words left are used as concept words. For example, in a comment like "I think this dress is beautiful and comfortable for a party", we can extract "dress", "beautiful", "comfortable", and "party". They represent the beautiful and comfortable property of the dress and the user's intention to wear the dress for a party. Since an item

[2] https://grouplens.org/datasets/movielens/
[3] http://jmcauley.ucsd.edu/data/amazon/
[4] https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data

Table 2
Statistics of the datasets.

Dataset	MovieLens 1M	Beauty	Steam	MovieLens 20M
#User	6040	40 226	281 428	138 493
#Item	3416	54 542	13 044	26 744
#Concept	100	90	117	81
#Concept Combination	1593	9316	8363	6461
#Popularity Level	10	10	10	10
#Interactions	999 611	353 962	3 488 885	20000263
Average length	165.4985	8.7993	12.3971	142.4135
Density	4. 8448%	0.0161%	0.0950%	0.5400%
Feature Sparsity of Graph	2.7127×10^{-4}	2.6742×10^{-5}	5.6878×10^{-5}	4.8484×10^{-5}

may contain one or more concept words, we use a combination of concept words to represent the attributes of the item, e.g., dress corresponds to the combination of concept words (“beautiful”, “comfortable”, “party”).

Table 2 summarizes the statistics of the pre-processed dataset, where # Concept and # Concept Combination stand for the count of concept words and concept word combinations, # Popularity Level stands for the number of different popularity levels, Average length denotes the average length of the interaction sequence, Density indicates the density of user–item interactions, and Feature Sparsity of Graph[5] indicates the sparsity of items and features based on feature hyperedges in the graph structure.

4.2. Experimental settings

We use the leave-one-out method to divide datasets. The last item, the penultimate item, and the remaining items of the interaction sequence are treated as the test item, the validation item, and the training set, respectively. To evaluate the performance of the model, for each user, we follow Sun et al. (2019) to randomly select 100 not-interacted items as negative samples for each test item, where the negative samples are sampled in the set of not-interacted items in accordance with their popularity. Our task is to rank this test item (i.e., the real item) and the 100 negative samples. We adopt the above setup for the evaluation of all models.

4.2.1. Baselines

To verify the effectiveness of IPSRec, we compare it with three groups of baseline approaches, which include non-sequential methods, sequential methods, and graph network-based methods.

Non-sequential methods:

- **PopRec**: A method sorts and recommends items for users based on their popularity, where the popularity is calculated by the frequency of the items’ occurrence in the sequence of all users.
- **BPR** (Rendle, Freudenthaler, Gantner, & Schmidt-Thieme, 2009)[6]: An approach for solving non-sequential Top-N recommendations, which performs Bayesian personalized ranking of items based on implicit user feedback.
- **NCF** (He et al., 2017)[7]: This approach uses a collaborative filtering approach based on neural networks to represent the potential interaction characteristics of users and items.

Sequential methods:

- **SASRec** (Kang & McAuley, 2018)[8]: A method models sequences through a self-attentive mechanism that captures the dependencies between items and predicts the next item to be recommended.
- **BERT4Rec** (Sun et al., 2019)[9]: A method that introduces the pre-trained BERT model to sequence recommendation and models the interaction sequences in both directions.
- **FDSA** (Vaswani et al., 2017)[10]: This method applies self-attention structure to item-level and feature-level sequences. It models item- and feature-transformation patterns separately and uses the learned representations for the next recommendation.
- **LOCKER** (He et al., 2021)[11]: In order to better capture short-term preferences, this method injects local constraints into the self-attentive network to enhance the learning of short-term information.

Graph-based methods:

[5] The density is calculated according to the formula $\rho = \frac{M}{\frac{1}{2}N(N-1)}$ for undirected graph networks, where M is the actual number of edges present and N is the number of nodes.

[6] https://github.com/sh0416/bpr

[7] https://github.com/hexiangnan/neural_collaborative_filtering

[8] https://github.com/kang205/SASRec

[9] https://github.com/FeiSun/BERT4Rec

[10] https://github.com/RUCAIBox/RecBole

[11] https://github.com/AaronHeee/LOCKER

- **HyperRec** (Wang et al., 2020)[12]: This is a method to truncate user interaction sequences based on timestamps and express short-term item relevance using sequence hypergraphs to enhance the next recommendation.
- **GES** (Zhu et al., 2023)[13]: This is a sequential recommendation method that considers the semantic relationship between items and builds the sequential adjacency graph and the semantic relation adjacency graph simultaneously.
- **HCCF** (Xia et al., 2022)[14]: This sequential recommendation method introduces hypergraph construction and local collaborative filtering, and enhances feature learning from two perspectives through contrastive learning.
- **DGSR** (Zhang, Wu, et al., 2022)[15]: A dynamic GNN sequence recommendation method that explores the interaction information of users and items with temporal and sequential information. It connects the interaction sequences of different users through dynamic graphs to achieve information propagation and aggregation in dynamic graphs.

We use the code provided by each paper to implement the above baseline methods. For the implementation of FDSA, we use the code provided by Zhao et al. (2021).

4.2.2. Evaluation metrics

We utilize three commonly used evaluation metrics in recommendation algorithms, hit rate HR@K, normalized discounted cumulative gain NDCG@K, and mean reciprocal rank MRR. Since each user has only one test item, we apply these metrics to each user in 101 items containing test item and negative samples. The results for all users are averaged. HR@K indicates the percentage of real items in the top K items for the users who participated in the test. NDCG@K assigns more weight to higher ranked items in the top K items, emphasizing that the target item should be ranked higher. MRR demonstrates whether the target items are placed in a more prominent position for the users. Larger values of these metrics indicate better performance of the recommendation algorithm. In our work, K is set to 1, 5 and 10. HR@1 is equal to NDCG@1, so we skip the calculation of NDCG@1.

4.2.3. Implementation details

We reuse the source code and recommended hyperparameters given by the authors of methods or the recommendation algorithm integration library for all baseline models. We use the Pytorch framework to implement IPSRec, using Adam for optimization, initial learning rate of 0.01, batch data size of 256, embedding size of 32, multiple attention layer heads of 4, and multiple attention layers stacked with 2 layers. For the MovieLens dataset, the count of layers of the graph network is 2, and the local attention head and global attention head are set to 1 and 3 in the multi-headed attention layer. For the Beauty dataset, the number of layers of the graph network is 1, and the local attention head and global attention head are set to 0 and 4 in the multi-headed attention layer. For the Steam dataset, the count of layers of the graph network is 1, and the local attention head and global attention head are set to 1 and 3 in the multi-headed attention layer. The maximum length of user interaction sequences is set according to the average length of all sequences and is set to 200 for the MovieLens dataset, 20 for the Beauty dataset, and 50 for the Steam dataset. The dropout rate is set to 0.2 for the attention layer of MovieLens, and 0.5 for Beauty and Steam. We perform a grid search to optimize the setting of hyperparameters by selecting l_2 regularization coefficients from $\{1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1\}$.

4.3. Overall performance comparison

We compare the performance of IPSRec with baseline methods. The results are demonstrated in Table 3. We elaborate on the analysis of the results in the below points.

(1) Non-sequential methods generally have poorer performance than sequential methods. PopRec, a recommendation method based on popularity ranking, only calculates and ranks the popularity of items, which considers the popular preferences but not users' personalized preferences. It does not consider the temporal information of user interactions and thus has the worst performance. BPR and NCF model the static preferences of users without factoring dynamic changes in user preferences and thus have poorer performance. These sequential approaches use the attention blocks to study user sequential behaviors. They learn sequential information about user interaction sequences considering item-to-item correlations. This shows that it is essential to consider sequential information to get better recommendation performance in the sequential recommendation scenario.

(2) The attention mechanism can significantly improve the sequence recommendation effect. SASRec and BERT4Rec model user interaction sequences based on the Transformer. They learn the influence of different items on the next prediction to capture the dynamic changes in user behaviors. LOCKER applies local constraints in the attention mechanism to capture the short-term intentions of users. The recommendation results show that applying the attention mechanism to model the transfer of user intentions and preferences can obtain better results. Particularly, the user's intentions are more obvious in the short-term migration in the interaction sequence. The locally constraint short-term learning is needed to obtain a more accurate representation of the user's short-term intentions.

[12] https://github.com/wangjlgz/HyperRec
[13] https://github.com/zhuty16/GES
[14] https://github.com/akaxlh/HCCF
[15] https://github.com/CRIPAC-DIG/DGSR

Table 3
Recommended performance of different methods.

Datasets	Metric	Non-sequential methods			Sequential methods				GNN-based methods					Improve
		PopRec	BPR	NCF	FDSA	SASRec	BERT4Rec	LOCKER	HyperRec	GES	HCCF	DGSR	IPSRec	
ml-1m	HR@1	0.0154	0.0799	0.0673	0.1939	0.2280	0.2561	0.2446	0.1394	0.2396	0.1133	0.2209	**0.2924**	**14.1742%**
	HR@5	0.0699	0.3062	0.2483	0.5161	0.5457	0.5687	0.4565	0.3985	0.5541	0.3555	0.5315	**0.5829**	**2.4969%**
	HR@10	0.1396	0.4862	0.3998	0.6505	0.6758	0.6802	0.5720	0.5496	0.6823	0.5156	0.6641	**0.6957**	**1.9639%**
	NDCG@5	0.0422	0.1913	0.1575	0.3615	0.3942	0.4223	0.3529	0.2717	0.4043	0.2376	0.3787	**0.4447**	**5.3043%**
	NDCG@10	0.0644	0.2492	0.2063	0.4052	0.4364	0.4587	0.3906	0.3208	0.4460	0.2868	0.4221	**0.4818**	**5.0360%**
	MRR	0.0642	0.2019	0.1721	0.3425	0.3751	0.4008	0.3489	0.2683	0.3853	0.2401	0.3627	**0.4262**	**6.3373%**
beauty	HR@1	0.0098	0.0428	0.0271	0.0239	0.0795	0.0551	0.0574	0.0451	0.0630	0.0781	0.095	**0.1084**	**14.1053%**
	HR@5	0.0444	0.1513	0.1166	0.0988	0.2004	0.1438	0.1326	0.1504	0.2305	0.1914	0.2363	**0.2562**	**8.4215%**
	HR@10	0.0840	0.2556	0.2145	0.1768	0.2728	0.1967	0.1900	0.2379	0.3365	0.3125	0.3226	**0.3490**	**3.7147%**
	NDCG@5	0.0268	0.0971	0.0711	0.0610	0.1417	0.1009	0.0956	0.0980	0.1463	0.1372	0.1672	**0.1845**	**10.3469%**
	NDCG@10	0.0395	0.1303	0.1024	0.0859	0.1650	0.1180	0.1139	0.1261	0.1867	0.1779	0.1946	**0.2143**	**10.1233%**
	MRR	0.0477	0.1184	0.0952	0.0819	0.1501	0.1114	0.1099	0.1134	0.1623	0.1589	0.1760	**0.1937**	**10.0568%**
steam	HR@1	0.0143	0.0331	0.0263	0.0307	0.0659	0.0755	0.0292	0.0398	0.0723	0.0781	0.1253	**0.1073**	**−14.3655%**
	HR@5	0.0680	0.1453	0.1263	0.1306	0.2289	0.2274	0.1047	0.1456	0.2500	0.1758	0.3115	**0.2903**	**−6.8058%**
	HR@10	0.1222	0.2627	0.2337	0.2327	0.3546	0.3408	0.1736	0.2411	0.3809	0.2969	0.4442	**0.4188**	**−5.7181%**
	NDCG@5	0.0407	0.0881	0.0752	0.0800	0.1481	0.1522	0.0669	0.0924	0.1617	0.1158	0.2201	**0.1993**	**−9.4502%**
	NDCG@10	0.0581	0.1257	0.1096	0.1127	0.1884	0.1886	0.0890	0.1231	0.2040	0.1521	0.2628	**0.2396**	**−8.8280%**
	MRR	0.0608	0.1123	0.0982	0.1022	0.1613	0.1637	0.0836	0.1111	0.1727	0.1377	0.2291	**0.2078**	**−9.2973%**
ml-20m	HR@1	0.0215	0.0435	0.0352	0.1458	0.1637	0.2323	0.2254	0.0760	0.1964	0.1133	0.1868	**0.2948**	**26.9049%**
	HR@5	0.0810	0.1696	0.1510	0.4245	0.4761	0.5047	0.3156	0.2569	0.5031	0.3438	0.4765	**0.5982**	**18.5259%**
	HR@10	0.1382	0.2999	0.2737	0.5847	0.6472	0.6327	0.5048	0.3949	0.6535	0.5352	0.6236	**0.7242**	**10.8187%**
	NDCG@5	0.0510	0.1056	0.0921	0.2882	0.3220	0.3745	0.3155	0.1670	0.3552	0.2183	0.3369	**0.4532**	**21.0147%**
	NDCG@10	0.0693	0.1474	0.1313	0.3400	0.3779	0.4160	0.3490	0.2113	0.4039	0.2742	0.3841	**0.4952**	**19.0385%**
	MRR	0.0707	0.1279	0.1147	0.2827	0.3117	0.3633	0.3177	0.1773	0.3421	0.2257	0.3268	**0.4359**	**19.9835%**

(3) Modeling sequential information using the structure of graphs has a positive influence on the results. Due to the graph structure, graph neural networks can fuse multiple information into a graph. Besides, they can obtain the representation of items and their attributes in different sequences by aggregating information from neighboring nodes for higher-order relationship modeling. HyperRec applies multiple types of convolutional layers based on sequence hypergraphs to capture high-order relationships in hypergraphs. Considering item semantic relations, GES constructs a semantic adjacency matrix in addition to a sequence adjacency matrix. HCCF introduces both the hypergraph learning module and local decomposition module, and constructs a contrastive learning task. DGSR connects interaction sequences of different users through dynamic graphs, and its recommendation results also verify the effectiveness of graph networks in sequence recommendation.

(4) Our method, IPSRec, outperforms the baseline methods on most evaluation metrics for our selected datasets. For example, on the metric HR@10, we improved 14.17% on the MovieLens 1M dataset, 14.11% on Beauty and 26.9049% on MovieLens 20M compared to the sub-optimal model. For denser long sequences like MovieLens, sequence-based methods can learn long-term dependencies within user interaction sequences for the next recommendation. However, graph-based methods like HyperRec and DGSR do not achieve better results. For sparse datasets with short sequences like Beauty, graph-based methods model higher-order correlations between nodes. Therefore, DGSR attains better results than sequential methods. IPSRec learns the sequential information within sequences based on Transformer while considering the higher-order connectivity between users, items, and auxiliary information. Therefore, it significantly improves compared to baseline methods in both long and short-sequence datasets.

(5) On the Steam dataset, we do not perform better than DGSR. The Steam dataset has a larger time span and fewer records of user interactions than other datasets. DGSR converts all user sequences into dynamic graphs and embeds temporal features at the edges of the dynamic graphs. Specifically, DGSR manually constructs a mapping between time (year and month) and number in the data processing part, and constructs subgraphs based on user sequences and time. Since DGSR considers time information when constructing dynamic graphs, it can explicitly embed time features when there is less historical user interaction data. As a result, it can learn items and users that are more relevant to the current interests of users, thus improving the performance when the data time span is large. DGSR models the time component with more manual processing and a more significant number of parameters, improving performance on steam datasets. Our approach achieves optimal results with fewer parameters on other datasets and sub-optimal results on steam datasets, as seen in Section 4.6. Our following research focuses on how to further effectively solve the problem of learning user intentions and preferences on sparse datasets.

4.4. Ablation study

To have a deeper understanding of the performance of the components of our proposed IPSRec, we conduct ablation experiments on MovieLens 1M, Beauty, and Steam datasets and design five variants according to the number of components from less to more. For a fair comparison, the structure of the heterogeneous hypergraph used in all methods is 1 layer. The locally constrained multi-headed self-attentive layer has 4 heads, in which the count of local attention heads is 4, and the number of normal attention heads is 0. The results of the experiments are listed in Table 4.

Table 4
Results of ablation experiments.

Dataset	Metric	SASRec	IPSRec-Add	IPSRec-HGNN	IPSRec-Separate	IPSRec-Local	IPSRec-Nopop	IPSRec	Improve
ml-1m	HR@1	0.2280	0.2465	0.2368	0.2510	0.2700	0.2848	**0.2924**	**18.6207%**
	HR@5	0.5457	0.5404	0.5194	0.5376	0.5649	0.5757	**0.5829**	**7.8645%**
	HR@1	0.6758	0.6586	0.6358	0.6581	0.6783	0.6884	**0.6957**	**5.6332%**
	NDCG@5	0.3942	0.3997	0.3837	0.4002	0.4251	0.4384	**0.4447**	**11.2584%**
	NDCG@10	0.4364	0.4380	0.4212	0.4391	0.4618	0.4749	**0.4818**	**10.0000%**
	MRR	0.3751	0.3824	0.3687	0.3844	0.4063	0.4203	**0.4262**	**11.4540%**
beauty	HR@1	0.0795	0.0962	0.0910	0.0969	0.0899	0.0949	**0.1012**	**5.1975%**
	HR@5	0.2004	0.2330	0.2237	0.2356	0.2182	0.2287	**0.2409**	**3.3906%**
	HR@10	0.2728	0.3216	0.3130	0.3272	0.3085	0.3186	**0.3315**	**3.0784%**
	NDCG@5	0.1417	0.1663	0.1585	0.1664	0.1533	0.1625	**0.1726**	**3.7883%**
	NDCG@10	0.1650	0.1950	0.1864	0.1946	0.1822	0.1915	**0.2007**	**2.9231%**
	MRR	0.1501	0.1762	0.1692	0.1766	0.1669	0.1735	**0.1826**	**3.6322%**
steam	HR@1	0.0659	0.0721	0.0783	0.0969	0.1034	0.0720	**0.1073**	**48.8211%**
	HR@5	0.2289	0.2226	0.2340	0.2745	0.2805	0.2302	**0.2903**	**30.4133%**
	HR@10	0.3546	0.3435	0.3520	0.3943	0.4118	0.3536	**0.4188**	**21.9214%**
	NDCG@5	0.1481	0.1448	0.1566	0.1862	0.1922	0.1514	**0.1993**	**37.6381%**
	NDCG@10	0.1884	0.1837	0.1943	0.2247	0.2346	0.1912	**0.2396**	**30.4300%**
	MRR	0.1613	0.1596	0.1699	0.1949	0.1922	0.1649	**0.2078**	**30.2005%**

- **IPSRec-Add**: This approach incorporates the item's corresponding auxiliary information **concept word combinations** and **popularity rankings** into the item features in the form of summation before inputting them into the attention block, i.e., the item features $\mathbf{e} = \mathbf{i} + \mathbf{p}_i + \mathbf{c'}_i$ for the input attention block. This model does not incorporate the components of heterogeneous hypergraphs, nor does it have a locally constrained attention layer and a mutual learning component. The architecture is similar to a variant of SASRec, where item features are fused and fed into two self-attentive layers and a feedforward neural network for training. The next item recommended for the user is ranked and selected by the likelihood score of the recommended item.
- **IPSRec-HGNN**: This method adds a **heterogeneous hypergraph** based on the IPSRec-Add method. The features of the heterogeneous hypergraph aggregate the item features and auxiliary information. Then, the features are fused by summation and sent to the self-attentive layer for training.
- **IPSRec-Separate**: This method introduces **different levels of feature capturing** based on IPSRec-HGNN for multiple kinds of heterogeneous information. After feature aggregation in the heterogeneous hypergraph, the method captures user behavior patterns from three feature levels: item features, concept word combinations features, and popularity ranking features. They are fed into the self-attentive layer for training. A recommendation popularity score is added to the likelihood score when making the next recommendation for the user.
- **IPSRec-Local**: This method introduces a locally-constrained self-attentive layer based on IPSRec-Separate to capture different levels of features from different perspectives. Specifically, the method inputs item sequences to the locally constrained self-attentive layer to capture user intentions. As for the remaining two sequences, they are sent to the normal self-attentive layer for user preferences learning.
- **IPSRec-Nopop**: This method **removes the popularity prediction module** from IPSRec. It only predicts the target item without predicting the popularity level of it.
- **IPSRec**: IPSRec is our final model, which is based on IPSRec-Local using a **mutual learning module** for the fusion of different intentions and preferences.

According to experiment results, IPSRec-Add promotes the recommendation performance compared to SASRec, which only considers the interaction information. The only difference between IPSRec-Add and SASRec is that IPSRec-Add incorporates auxiliary information into the item's representations before using self-attentive training. This shows that the auxiliary information can improve the recommendation performance.

IPSRec-HGNN adds heterogeneous hypergraphs to the sequence-based model for propagation and aggregation among heterogeneous information. However, the results of only adding heterogeneous hypergraphs are degraded, which could be due to the noise and redundancy generated by the node representation after fusing neighboring features. The processing of node representations after fusing higher-order information should be handled separately from different perspectives based on the graph structure. In this situation, the semantics of node features will not be destroyed, and thus, better recommendation performance can be obtained. Noticing this, IPSRec-Separate learns three types of node information separately based on sequential hyperedges in the attention layer and obtains better performance than IPSRec-Add and IPSRec-HGNN.

The sparsity of user–item interaction sequences, concept word combinations sequences, and popularity ranking sequences are different. Besides, the user's item selection combines the user's preferences for concept word combinations, preferences for popularity levels, and specific intentions for the item. Taking the above into account, IPSRec-Local additionally uses a self-attentive component with local constraints to study the user's intentions for items. On MovieLens 1M and Steam, IPSRec-Local performs better than IPSRec-Separate. While on Beauty, the performance decreases instead. This situation is because MovieLens 1M and Steam are mainly long sequences. Using the locally constraint self-attention mechanism can better capture the user's short-term

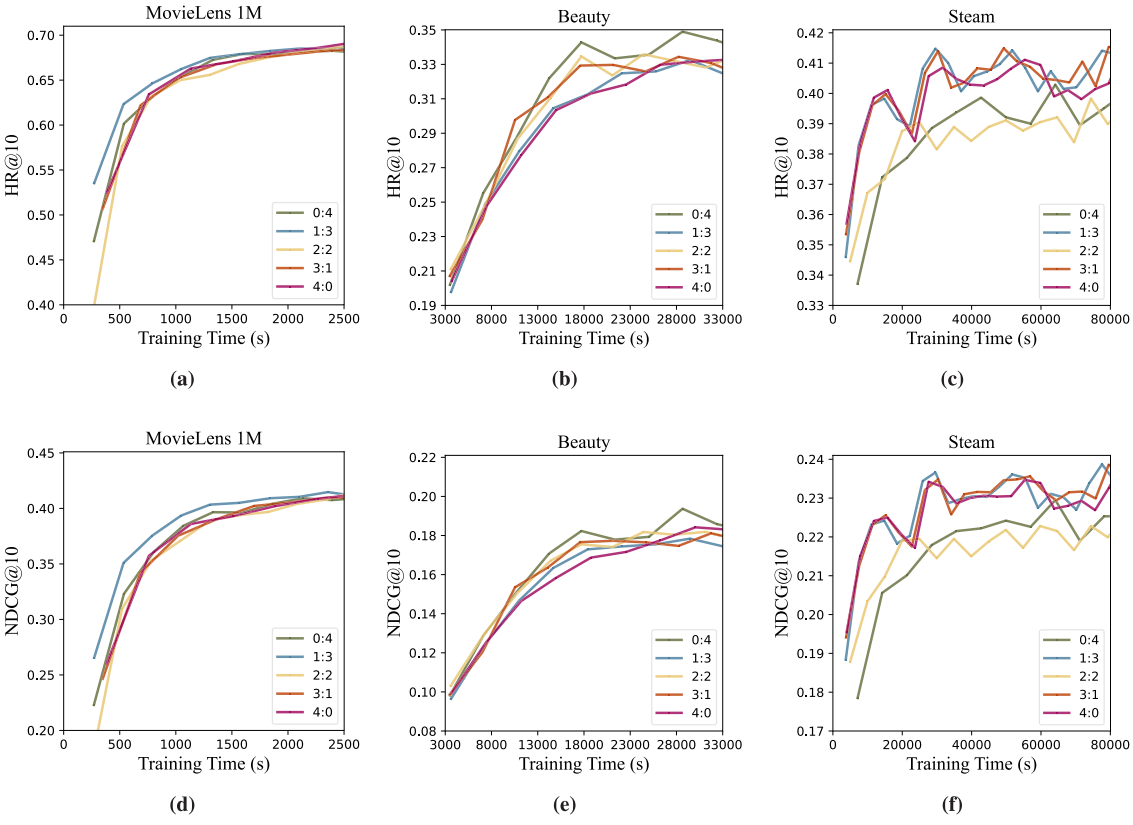

Fig. 3. Results at HR@10 and NDCG@10 for different number of local attention heads and normal attention heads in a locally constrained multi-headed attention layer.

intentions, thus improving recommendation accuracy. Beauty has a shorter average sequence length. Therefore, a normal attention mechanism can learn the user's intentions well. Using a distance-aware local attention mechanism increases noisy data and reduces the recommendation effect. Thus, for a short sequence dataset like Beauty, we can reduce the count of local attention heads and use more normal attention heads, which is also verified in Section 4.5.1.

IPSRec uses a mutual learning module based on IPSRec-Local to fuse user intentions and preferences. It can simulate the determination of users and improve the recommendation performance. It can be verified that the mutual learning module based on the gating mechanism effectively fuses multiple user intentions and preferences to improve recommendation accuracy. In addition, IPSRec-Nopop removes the popularity prediction module, resulting in significant performance degradation on multiple datasets. It indicates that simulating users' popularity-level preferences is helpful in modeling user preferences more accurately and improving the accuracy of recommendations.

4.5. Hyperparameters analysis

Subsequently, we verify the effect of different hyperparameters on the structure of locally constrained multi-headed attention layers and the number of layers of heterogeneous hypergraphs on MovieLens-1M, Beauty and Steam.

4.5.1. Effect of the number of local attention heads

In IPSRec, the count of heads of all multi-headed attention layers is set to 4. For the locally constrained multi-headed attention layer, the numbers of local attention heads and normal attention heads are set as l and $4-l$, respectively. We denote this structure as $l:4-l$. We study the effectiveness of the model on different datasets with different numbers of local attention heads and normal attention heads in the locally constrained multi-headed attention layer with $\{0:4, 1:3, 2:2, 3:1, 4:0\}$. The results are shown in Fig. 3. Since all these metric results showed similar upward or downward trends in the experiments, we chose the metric results of HR@10 and NDCG@10 for illustration and reported the results after every 20 epochs of model training.

For MovieLens-1M and Steam, the models structured with one local attention head and three normal attention heads achieve better recommendation performance while converging faster. When capturing users' intentions on items from the perspective of item interaction, using locally reinforced attention enhances learning of the short-term users' intentions. However, it also requires considering long-term historical user preferences, so normal attention is also needed to learn the long-term dependencies of

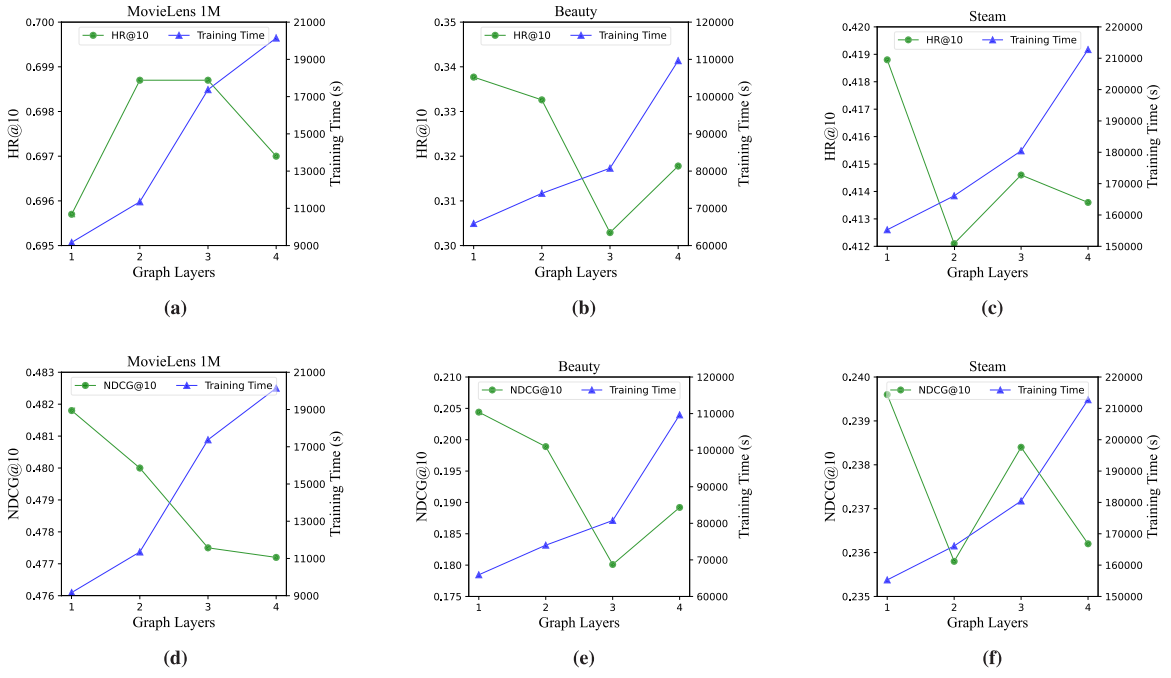

Fig. 4. Results of different heterogeneous hypergraph layers in HR@10 and NDCG@10.

sequences. For the Beauty dataset, the model structured with 0 local attention heads and 4 normal attention heads performs best. Because Beauty is a short sequence dataset with an average sequence length of 8.8, and the normal attention heads can learn the user's short-term intentions well enough. Adding local attention heads may generate unnecessary noise to destroy the original semantics and decrease the recommendation accuracy.

4.5.2. *Effect of the number of layers of heterogeneous hypergraphs*

Choosing the model with optimal performance requires considering accuracy and training time. We set up experiments to verify the influence of the number of layers of the heterogeneous hypergraph on accuracy and training time. We select the HR@10 and NDCG@10 metrics to report the results, which are shown in Fig. 4. The count of layers of the graph controls how well the nodes fuse the features of neighboring nodes. Stacking more layers of the hypergraph network enables nodes to obtain more information from neighboring nodes. But further increases in the number of layers may cause overfit. Besides, the model may obtain irrelevant information from multi-hop neighbors and over-smooth, decreasing the recommendation accuracy instead. As the count of heterogeneous hypergraph layers increases, the model gets larger size and the training time gets longer. For MovieLens-1M, the features are denser, so the multi-layer graph structure can learn more neighbor node information. IPSRec performs better with 2 layers of graph network, as the NDCG decreases slightly and time consumption is affordable. For Beauty and Steam, the features are sparse. Therefore, too many layers of the graph network will generate noise and its graph network get better results with 1 layer.

4.6. *Model complexity analysis*

Algorithm 1 shows the overall training structure of our model. Since heterogeneous hypergraphs can be constructed before the model training, we only analyze the time complexity during model training here. In the hypergraph information propagation module for feature hyperedges, the time complexity of intra aggregation for item nodes is $O(Md)$, where M is the number of all items and d is the vector dimension. The time complexity of side information is $O((Q+R)d)$, where Q is the number of all popularity levels and R is the number of all concept word combinations. As for inter aggregation module, its time complexity is $O((M+R+Q)d)$. Therefore, the time complexity of the whole hypergraph information propagation module is $O(L(M+R+Q)d)$, where L is the number of hypergraph layers. Since Q is set to 10 in each dataset, the time complexity can be approximately reduced to $O(L(M+R)d)$. In the attention learning module based on sequence hyperedges, the time complexity of the two attention mechanisms is $O(N_u L_a T^2 d)$, where T is the length of the sequence, N_u is the number of all user sequences, and L_a is the number of attention layers. For mutual learning modules, the time complexity is $O(N_u T^2 d^2)$. Considering all the above modules, the time complexity of the whole model is $O(L(M+R)d + N_u L_a T^2 d + N_u T d^2)$.

In terms of space complexity, we also conduct a comprehensive analysis of each module. In the hypergraph construction phase, there is a need for $O((M+R+Q)d)$ space to accommodate vectors and an additional $O(MR+QR+MQ)$ space to handle the adjacency matrix storage. In the hypergraph information propagation module, no additional parameter space is necessary for item

Table 5
Model parameters comparisons.

Method	HyperRec	GES	HCCF	DGSR	IPSRec
Size(MB)	6.9688	0.8967	1.1855	2.1278	1.0832

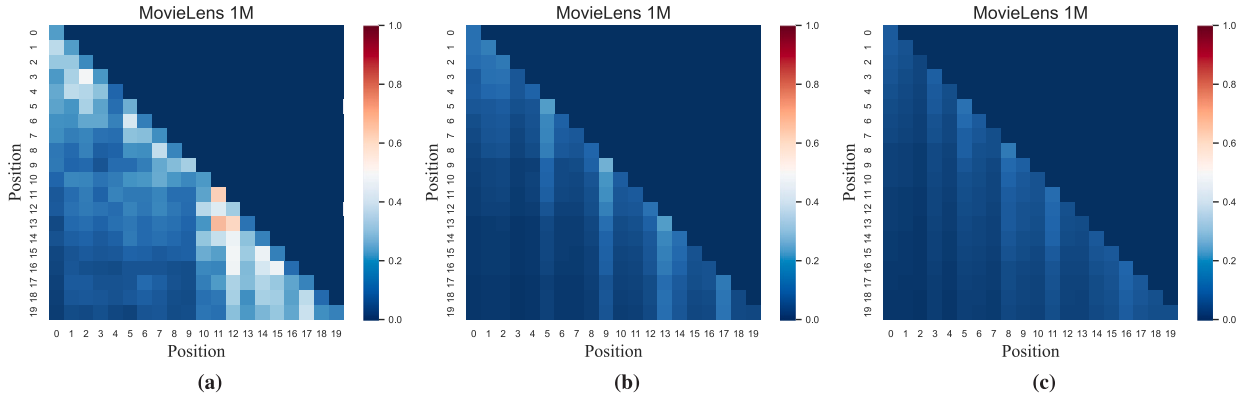

Fig. 5. Attention heatmaps of items, popularity levels, and concept word combinations sequences.

nodes in intra aggregation, and $O((R+Q)d)$ space is required for side information nodes to store the corresponding learnable vectors; $O(d^2)$ space is needed for inter aggregation. Within the attention learning module, $O(Td)$ space is allocated to store the position vectors. Long-term preferences learning demands $O(L_a d^2)$ space for storing the parameters of the multi-head attention mechanism and $O(L_a(d^2+d))$ for the parameters of the FFN. For short-term intentions learning, it necessitates $O(Td)$ space for the distance matrix, $O(L_a(d^2+d))$ space for the locally reinforced attention, and $O(L_a(d^2+d))$ for the parameters of the FFN. For the mutual learning module, $O(d^2)$ space is required to store the parameters. Combining all the above modules, the space complexity of the whole model is $O((M+R+Q)d+MR+QR+MQ+(R+Q)d+d^2+Td+L_a d^2+L_a(d^2+d)+Td+2L_a(d^2+d)+d^2)$. Similar to the time complexity analysis module, we simplify the constant level term to a space complexity of $O(M+R+MR+(M+R+T+L_a)d+L_a d^2)$.

Since the analysis methods on time and space complexity vary across these papers, we count the parameters used by several state-of-the-art methods to determine when they get optimal results on the MovieLens-1 m dataset for convenient comparison. From Table 5, it can be seen that our method achieves the optimal results on the dataset with a smaller number of parameters. We can conclude that IPSRec is advanced in terms of model training consumption.

Algorithm 1: The training procedure for IPSRec

Input : The interaction sequence $\mathcal{I}_{\mathcal{U}}$, the popularity ranking sequence $\mathcal{P}_{\mathcal{U}}$ and the combinations of concept word sequence $\mathcal{C}'_{\mathcal{U}}$, epoch number N_e, batch number N_b

Initialize $\mathbf{E}^I$, $\mathbf{E}^P$, $\mathbf{E}^{C'}$ randomly ;
Construct heterogeneous hypergraph $\mathcal{G}=(\mathcal{V},\mathcal{E},\mathcal{A})$ as Section 3.4 ;
for $i \leftarrow 1$ **to** N_e **do**
  $\mathbf{E}^I$, $\mathbf{E}^P$, $\mathbf{E}^{C'} \leftarrow$Message Propagation `// Update nodes by aggregation of hypergraph` ;
  Sample a mini batch from $\mathcal{I}_{\mathcal{U}}$, $\mathcal{P}_{\mathcal{U}}$ and $\mathcal{C}'_{\mathcal{U}}$;
  for $j \leftarrow 1$ **to** N_b **do**
    $\widehat{\mathbf{P}^u}, \widehat{\mathbf{C}'^u} \leftarrow$ Long-term Preferences Learning `// Get sequence embedding of preferences` ;
    $\widehat{\mathbf{I}^u} \leftarrow$ Short-term Intentions Learning `// Get sequence embedding of intentions` ;
    $\mathbf{E}_{IP}$, $\mathbf{E}_{PI}$, $\mathbf{E}_{C'I} \leftarrow$ Mutual Learning Schema `// Combine preferences and intentions`;
    Calculate loss as Eq.(39) and update model parameters ;
  end
end
Output: IPSRec model

4.7. Visualizing attention weights

The attention heatmaps after the Attention Mechanism module are shown in Fig. 5. They are the attention heatmaps of item sequences, popularity level sequences, and concept word combinations sequences of the user u from MovieLen-1M dataset corresponding to Fig. 5(a), Fig. 5(b), and Fig. 5(c), respectively. In order to show the heatmaps clearly, we intercepted 20 recent nodes in the sequences. The attention mechanism module in our model is used to learn the user's short-term intentions and long-term

preferences differentially. Specifically, we adopt a locally reinforced attention mechanism to pay more attention to the user's short-term intentions for the item sequence hyperedges. For the popularity level and concept word combinations sequence hyperedge, we use a normal multi-head attention mechanism to pay more attention to the user's long-term preferences. As can be seen from the three graphs, the attention heatmap of the user u's item sequences pays more attention to items closer to the current one, which is in line with our approach for the user's short-term intention. As for the user's popularity level and concept word combination sequences, the model more evenly concerns the nodes over a long time due to the normal multi-attention mechanism, which is a simulation of the user's long-term preference in the sequences.

From Fig. 5(b), we can see that items 5, 9, 13, and 17 have significantly higher attention weights. As we observe and analyze the original popularity sequence, these items mainly correspond to the popularity level of "7". From this, we can see that user u favors movies in this popularity level over a longer period, effectively reflecting user u's long-term preference in the popularity level.

Our model comprehensively considers the short-term intentions and long-term preferences of users. Besides, we differentiate their various degrees of deviation in the user sequence in the attention mechanism module, which can effectively simulate the real intentions and preferences of users. Thus, IPSRec predicts the behavioral decisions of users in a reasonable way and improves accuracy.

5. Discussion

5.1. Results analysis

Our proposed IPSRec aims to fuse diverse heterogeneous auxiliary information, explicitly mine users' short-term intentions and long-term preferences, and model their migration in sequences. Comparing IPSRec with state-of-the-art sequence recommendation methods, experiment results show its superiority. Specifically, in the overall comparison, IPSRec better characterizes users' short-term intentions and long-term preferences in sequences, outperforming baseline approaches on most evaluation metrics across all datasets. According to the results of the ablation experiments, the diverse heterogeneous auxiliary information, the heterogeneous hypergraph feature aggregation module, the attention mechanism module that introduces local constraints, the gate-based mutual learning module, and the popularity predicting all contribute to the model performance improvement. In addition, through the tuning of hyperparameters, IPSRec can achieve excellent results on datasets with different sparsity levels while maintaining acceptable time consumption. Furthermore, according to the model complexity analysis, IPSRec has advantages in terms of training cost. By comparing our IPSRec with these baseline models, the main advantages of our work are mentioned below: (1) By fusing diverse heterogeneous auxiliary information and aggregating features using hypergraph modules, our proposed model can effectively explicitly model user intentions and user preferences. (2) The attention mechanism module that introduces local constraints can learn the migration of users' short-term intentions and long-term preferences in sequences through different attention learning. (3) Each independent module of IPSRec works efficiently together, which intuitively explains why our model achieves better results than other approaches.

5.2. Theoretical and practical implications

Considering the theoretical perspective, IPSRec explicitly mines users' short-term intentions and long-term preferences by fusing diverse heterogeneous auxiliary information, which helps to better model user–item interaction sequences in sequence recommendation scenarios. First, IPSRec introduces item-level intentions representation, concept-level preferences representation, and popularity-level preferences representation. Secondly, for the potential topological links existing between them, IPSRec constructs a heterogeneous hypergraph module for information propagation and feature aggregation of different types of auxiliary information. Subsequently, we use different attention mechanisms with multiple perspectives to learn the migration of users' short-term intentions and long-term preferences in sequences. Furthermore, we perform the fusion of different levels of user intentions and preferences. Experiment results show that IPSRec achieves excellent results on multiple datasets and outperforms existing sequence recommendation methods.

In the sequential recommendation scenario, there are only user–item interaction sequences. Compared to the traditional recommendation scenario, the implicit feedback information, such as user personal information and browsing history information, is lacking. Traditional recommendation methods that utilize user information and mine implicit feedback perform poorly in this scenario. Therefore, this paper focuses on how to integrate diverse heterogeneous auxiliary information in user–item interaction sequences, explicitly mine users' short-term intentions and long-term preferences, achieve more accurate representations of users' real intentions and preferences, and make more reliable next-item recommendations to achieve higher accuracy.

6. Conclusion

We propose IPSRec, a sequential recommendation approach for explicitly mining users' short-term intentions and long-term preferences by fusing heterogeneous auxiliary information from multiple sources. To achieve this, IPSRec constructs a heterogeneous hypergraph to explicitly model user intentions and preferences features as well as their potential joint effect. A locally reinforced attention block is further utilized to differentially learn the migration of users' short-term intentions and long-term preferences across interaction sequences. Experiments conducted on various datasets show that our model obtains better performance compared to state-of-the-art approaches.

We hope this paper can demonstrate a new perspective on portraying user behavior in sequence recommendation. Although IPSRec takes some auxiliary information into account, the data sparsity problem still has severe influence in real scenarios, which causes performance degradation in some datasets. In the future, we will attempt to improve the construction of hypergraphs that fuse heterogeneous information, such as learning popularity level preferences at a finer granularity. For the mining and use of conceptual words, we will introduce semantic information from the pre-trained language model to obtain more natural language-informative representations. In addition, how to utilize other auxiliary information in more sparse data scenarios also deserves further research.

CRediT authorship contribution statement

Bingqian Liu: Conceptualization, Methodology, Writing – original draft. **Duantengchuan Li:** Conceptualization, Methodology, Writing – original draft. **Jian Wang:** Conceptualization, Methodology, Writing – review & editing. **Zhihao Wang:** Conceptualization, Investigation, Writing – original draft. **Bing Li:** Supervision, Writing – review & editing. **Cheng Zeng:** Writing – review & editing.

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

Data will be made available on request.

Acknowledgments

This work is supported by the Ministry of Science and Technology of the People's Republic of China "Science and Technology Innovation 2030" (No. 2021ZD0113405), the National Natural Science Foundation of China (No. 62032016), the Key Research and Development Program of Hubei Province (No. 2021BAA031), and the Foundation of Yunnan Key Lab of Service Computing (No. YNSC23102). Corresponding author are Jian Wang, Bing Li and Cheng Zeng.

References

Chang, J., Gao, C., Zheng, Y., Hui, Y., Niu, Y., Song, Y., ... Li, Y. (2021). Sequential recommendation with graph neural networks. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval* (pp. 378–387).

Devooght, R., & Bersini, H. (2017). Long and short-term recommendations with recurrent neural networks. In *Proceedings of the 25th conference on user modeling, adaptation and personalization* (pp. 13–21).

Du, Y., Peng, Z., Niu, J., & Yan, J. (2022). A unified hierarchical attention framework for sequential recommendation by fusing long and short-term preferences. *Expert Systems with Applications*, *201*(C).

Fan, S., Zhu, J., Han, X., Shi, C., Hu, L., Ma, B., et al. (2019). Metapath-guided heterogeneous graph neural network for intent recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2478–2486).

Feng, Y., You, H., Zhang, Z., Ji, R., & Gao, Y. (2019). Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence, vol. 33, no. 01* (pp. 3558–3565).

Han, J., Tao, Q., Tang, Y., & Xia, Y. (2022). DH-HGCN: Dual homogeneity hypergraph convolutional network for multiple social recommendations. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 2190–2194).

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, *5*(4), 1–19.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173–182).

He, R., & McAuley, J. (2016a). Fusing similarity models with Markov chains for sparse sequential recommendation. In *2016 IEEE 16th international conference on data mining* (pp. 191–200).

He, R., & McAuley, J. (2016b). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th international conference on world wide web* (pp. 507–517).

He, Z., Zhao, H., Lin, Z., Wang, Z., Kale, A., & Mcauley, J. (2021). Locker: Locally constrained self-attentive sequential recommendation. In *Proceedings of the 30th ACM international conference on information & knowledge management* (pp. 3088–3092).

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based recommendations with recurrent neural networks. CoRR abs/1511.06939.

Kang, W.-C., & McAuley, J. (2018). Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining* (pp. 197–206).

Kenton, J. D. M.-W. C., & Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT, vol. 1* (p. 2).

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

Li, Y., Chen, H., Sun, X., Sun, Z., Li, L., Cui, L., ... Xu, G. (2021). Hyperbolic hypergraphs for sequential recommendation. In *Proceedings of the 30th ACM international conference on information & knowledge management* (pp. 988–997).

Li, D., Deng, C., Wang, X., Li, Z., Zheng, C., Wang, J., et al. (2024). Joint inter-word and inter-sentence multi-relation modeling for summary-based recommender system. *Information Processing & Management*, *61*(3), Article 103631.

Li, Y., Gao, C., Luo, H., Jin, D., & Li, Y. (2022). Enhancing hypergraph neural networks with intent disentanglement for session-based recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 1997–2002).

Li, H., Wang, J., Du, X., Hu, Z., & Yang, S. (2023). KBHN: A knowledge-aware bi-hypergraph network based on visual-knowledge features fusion for teaching image annotation. *Information Processing & Management*, *60*(1), Article 103106.

Li, H., Wang, X., Zhang, Z., Ma, J., Cui, P., & Zhu, W. (2021). Intention-aware sequential recommendation with structured intent transition. *IEEE Transactions on Knowledge and Data Engineering*, *34*(11), 5403–5414.

Li, D., Xia, T., Wang, J., Shi, F., Zhang, Q., Li, B., et al. (2024). SDFormer: A shallow-to-deep feature interaction for knowledge graph embedding. *Knowledge-Based Systems*, *284*, Article 111253.
Li, Z., Zhang, Q., Zhu, F., Li, D., Zheng, C., & Zhang, Y. (2023). Knowledge graph representation learning with simplifying hierarchical feature propagation. *Information Processing & Management*, *60*(4), Article 103348.
Liu, C., Li, X., Cai, G., Dong, Z., Zhu, H., & Shang, L. (2021). Noninvasive self-attention for side information fusion in sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 5* (pp. 4249–4256).
Liu, Y., Li, D., Wang, J., Li, B., & Hang, B. (2024). MDLR: A multi-task disentangled learning representations for unsupervised time series domain adaptation. *Information Processing & Management*, *61*(3), Article 103638.
Liu, H., Zheng, C., Li, D., Shen, X., Lin, K., Wang, J., ... Xiong, N. N. (2022). EDMF: Efficient deep matrix factorization with review feature learning for industrial recommender system. *IEEE Transactions on Industrial Informatics*, *18*(7), 4361–4371.
McAuley, J., Targett, C., Shi, Q., & Van Den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval* (pp. 43–52).
Pathak, A., Gupta, K., & McAuley, J. (2017). Generating and personalizing bundle recommendations on steam. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval* (pp. 1073–1076).
Quadrana, M., Karatzoglou, A., Hidasi, B., & Cremonesi, P. (2017). Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the eleventh ACM conference on recommender systems* (pp. 130–137).
Rashed, A., Elsayed, S., & Schmidt-Thieme, L. (2022). Context and attribute-aware sequential recommendation via cross-attention. In *Proceedings of the 16th ACM conference on recommender systems* (pp. 71–80).
Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence* (pp. 452–461).
Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2010). Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on world wide web* (pp. 811–820).
Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, 1–20.
Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295).
Shen, X., Yi, B., Liu, H., Zhang, W., Zhang, Z., Liu, S., et al. (2021). Deep variational matrix factorization with knowledge embedding for recommendation system. *IEEE Transactions on Knowledge and Data Engineering*, *33*(5), 1906–1918.
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.
Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., et al. (2019). BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 1441–1450).
Tang, J., & Wang, K. (2018). Personalized top-N sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining* (pp. 565–573).
Tuan, T. X., & Phuong, T. M. (2017). 3D convolutional networks for session-based recommendation with content features. In *Proceedings of the eleventh ACM conference on recommender systems* (pp. 138–146).
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 6000–6010).
Wan, M., & McAuley, J. (2018). Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM conference on recommender systems* (pp. 86–94).
Wang, J., Ding, K., Hong, L., Liu, H., & Caverlee, J. (2020). Next-item recommendation with sequential hypergraphs. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 1101–1110).
Wang, J., Ding, K., Zhu, Z., & Caverlee, J. (2021). Session-based recommendation with hypergraph attention networks. In *Proceedings of the 2021 SIAM international conference on data mining* (pp. 82–90).
Wang, S., Hu, L., Wang, Y., Cao, L., Sheng, Q. Z., & Orgun, M. (2019). Sequential recommender systems: Challenges, progress and prospects. In *Proceedings of the twenty-eighth international joint conference on artificial intelligence* (pp. 6332–6338).
Wang, J., Zhang, Q., Shi, F., Li, D., Cai, Y., Wang, J., ... Zheng, C. (2023). Knowledge graph embedding model with attention-based high-low level features interaction convolutional network. *Information Processing & Management*, *60*(4), Article 103350.
Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J., & Jing, H. (2017). Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining* (pp. 495–503).
Wu, B., He, X., Wu, L., Zhang, X., & Ye, Y. (2023). Graph-augmented co-attention model for socio-sequential recommendation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *53*(7), 4039–4051.
Wu, F., Li, D., Lin, K., & Zhang, H. (2021). Efficient nodes representation learning with residual feature propagation. In *Advances in knowledge discovery and data mining* (pp. 156–167).
Wu, B., Zhong, L., Yao, L., & Ye, Y. (2022). EAGCN: An efficient adaptive graph convolutional network for item recommendation in social internet of things. *IEEE Internet of Things Journal*, *9*(17), 16386–16401.
Xia, L., Huang, C., Xu, Y., Zhao, J., Yin, D., & Huang, J. (2022). Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 70–79).
Xia, X., Yin, H., Yu, J., Wang, Q., Cui, L., & Zhang, X. (2021). Self-supervised hypergraph convolutional networks for session-based recommendation. In *Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 5* (pp. 4503–4511).
Xie, Y., Zhou, P., & Kim, S. (2022). Decoupled side information fusion for sequential recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 1611–1621).
Xu, C., Feng, J., Zhao, P., Zhuang, F., Wang, D., Liu, Y., et al. (2021). Long- and short-term self-attention network for sequential recommendation. *Neurocomputing*, [ISSN: 0925-2312] *423*, 580–589.
Yang, F., Li, G., & Yue, Y. (2022). Hierarchical user intention–preference for sequential recommendation with relation-aware heterogeneous information network embedding. *Big Data*, *10*(5), 466–478.
You, J., Wang, Y., Pal, A., Eksombatchai, P., Rosenburg, C., & Leskovec, J. (2019). Hierarchical temporal convolutional networks for dynamic recommender systems. In *The world wide web conference* (pp. 2236–2246).
Zhang, J., Wang, D., & Yu, D. (2021). TLSAN: Time-aware long- and short-term attention network for next-item recommendation. *Neurocomputing*, *441*, 179–191.
Zhang, M., Wu, S., Yu, X., Liu, Q., & Wang, L. (2022). Dynamic graph neural networks for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *35*(5), 4741–4753.
Zhang, X., Xu, B., Yang, L., Li, C., Ma, F., Liu, H., et al. (2022). Price does matter! modeling price and interest preferences in session-based recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 1684–1693).
Zhang, T., Zhao, P., Liu, Y., Sheng, V. S., Xu, J., Wang, D., et al. (2019). Feature-level deeper self-attention network for sequential recommendation. In *IJCAI* (pp. 4320–4326).

Zhao, W. X., Mu, S., Hou, Y., Lin, Z., Chen, Y., Pan, X., et al. (2021). Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th acm international conference on information & knowledge management* (pp. 4653–4664).

Zheng, Y., Gao, C., Chang, J., Niu, Y., Song, Y., Jin, D., et al. (2022). Disentangling long and short-term interests for recommendation. In *Proceedings of the ACM web conference 2022* (pp. 2256–2267).

Zhu, N., Cao, J., Liu, Y., Yang, Y., Ying, H., & Xiong, H. (2020). Sequential modeling of hierarchical user intention and preference for next-item recommendation. In *Proceedings of the 13th international conference on web search and data mining* (pp. 807–815).

Zhu, T., Sun, L., & Chen, G. (2023). Graph-based embedding smoothing for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering, 35*(1), 496–508.