Contents lists available at ScienceDirect

Expert Systems With Applications

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

Session-based recommendation with hypergraph convolutional networks and sequential information embeddings

Chengxin Ding [a], Zhongying Zhao [a,*], Chao Li [b,*], Yanwei Yu [c], Qingtian Zeng [a]

[a] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[b] College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[c] College of Computer Science and Technology, Ocean University of China, Qingdao, Shandong 266100, China

ARTICLE INFO

Keywords:
Recommendation
Session-based recommendation
Hypergraph convolutional network
Sequential information
Soft attention mechanism

ABSTRACT

Session-based recommendation focuses on predicting the next item that an anonymous user is most likely to click. Due to its privacy-protecting ability, it is receiving increasing attention from researchers in recent years. The existing studies typically focus on sequence or graph structure learning. However, they ignore the consistency and sequential dependence relationships that are widely existed between items in real-world scenarios. To address this problem, we present a novel method named HyperS^{2}Rec. Specifically, we propose to learn two kinds of item embeddings with hypergraph convolutional network and gated recurrent unit, respectively, to account for both consistency-awareness and sequential dependence-awareness. Then the attention mechanism is designed to flexibly combine the above both embeddings. Finally, the reversed position and the soft attention mechanism are utilized to obtain session representations. To verify the effectiveness of the proposed HyperS^{2}Rec, we conduct experiments on three real-world datasets. The results prove that the proposed HyperS^{2}Rec significantly outperforms state-of-the-art methods. The source code of our proposed model is available at https://github.com/ZZY-GraphMiningLab/HyperS2Rec.

1. Introduction

With the rapid growth of the Internet, users are flooded with plenty of information. Recommendation systems aim to provide users with the information they need in a timely and effective manner (Yue, Xiao, Zhao, & Li, 2022; Zhao, Yang, Li, & Nie, 2021; Zhao et al., 2020), and have become an essential technology to alleviate information overload (Malik, Rana, & Bansal, 2020; Zhang, Yao, Sun, & Tay, 2019; Zhao et al., 2022). Traditional studies (Aggarwal, 2016; Burke, 2002) usually rely on long-term historical interactions and complete user profiles to accomplish good performance. However, in real-world recommendation scenarios, user profiles are generally not available (Guo et al., 2019). For example, if a user does not log in online shopping websites when browsing items, only short-term historical interactions in the ongoing session can be used to predict the next click (Latifi, Mauro, & Jannach, 2021). Thus, it is quite difficult for traditional methods to achieve the desired performance in such scenarios.

Significantly different from the early research, session-based recommendation is proposed to deal with this dilemma. It is able to provide anonymous users with recommending results by only considering their short-term historical interactions. Therefore, it is receiving increasing attention from researchers (Tan, Xu, & Liu, 2016; Wang, Lou, & Jiang, 2022; Zhang, Zheng et al., 2022). One of the most critical issue in session-based recommendation is how to accurately and efficiently capture and learn complex transitions of items from the limited information. As shown in Fig. 1(a), u_1 clicked accessories such as *phone cases*, *screen protectors* or *earphones* after clicking a *mobile phone*. We can see that the items in session 1 are sequentially dependent. Based on such dependent assumption, recurrent neural network (RNN) is widely applied due to its capability of learning sequential data. GRU4Rec (Hidasi, Karatzoglou, Baltrunas & Tikk, 2016) is the first work to employ RNN in the field of session-based recommendation. It innovatively regards the clicks of a user as a sequence. NARM (Li et al., 2017) captures the sequential behavior of users and the main purpose of a session via a hybrid encoder. STAMP (Liu, Zeng, Mokhosi, & Zhang, 2018) proposes to capture the short-term interest through the most recently clicked item. Although RNN-based methods mentioned above are able to well handle sequential information in a session, the strongly dependent assumption makes them fail to model the other important relationship, e.g., consistency. Consistency means that there is no strict order between items. As an example shown in Fig. 1(b), u_2 clicked a variety of

* Corresponding authors.
E-mail addresses: 1556416359@qq.com (C. Ding), zyzhao@sdust.edu.cn (Z. Zhao), lichao@sdust.edu.cn (C. Li), yuyanwei@ouc.edu.cn (Y. Yu), qtzeng@sdust.edu.cn (Q. Zeng).

https://doi.org/10.1016/j.eswa.2023.119875
Received 1 October 2022; Received in revised form 1 March 2023; Accepted 12 March 2023
Available online 15 March 2023

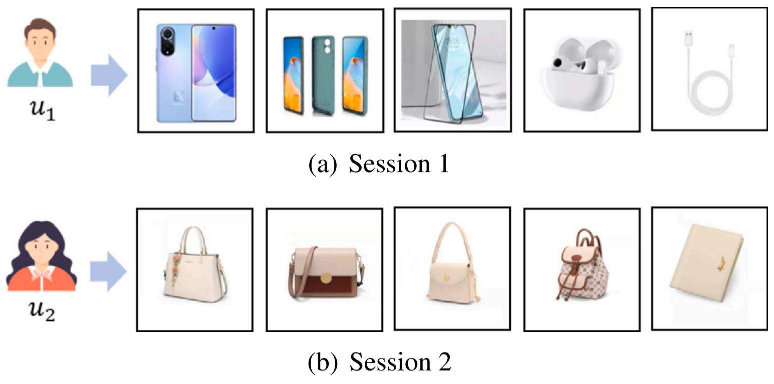

Fig. 1. A toy example of two kinds of relationships between items. From Fig. 1(a), we can see that the items in session 1 are sequentially dependent. While the items in session 2 are consistent as shown in Fig. 1(b). Both of the above relationships are widely existed in real-world scenarios and should be paid attention to.

bags, like *handbag, backpack, shoulder bag*, etc. Obviously, there is no sequential dependence but consistency between items in session 2. As the consistency is also widely existed in real-world scenarios, it is vital to capture both of relationships for session-based recommendation.

Graph neural network (GNN) based methods aim to model sessions as graph-structured data, which enables them to make full use of graph structure to enhance the recommending performance. SR-GNN (Wu, Tang et al., 2019) first attempts to model each session as a subgraph. It learns item representations via gated graph neural network (GGNN). GC-SAN (Xu et al., 2019) utilizes both GGNN and self-attention mechanism to enrich contextualized representations of items. SHARE (Wang, Ding, Zhu, & Caverlee, 2021) is designed based on hypergraph attention network (HGAT) and learns contextual information with sliding windows. DHCN (Xia et al., 2021) captures the beyond-pairwise relations among items through hypergraph modeling. Although GNN-based methods are effective in learning complex graph-structured transitions via the information propagation on the graph, they relax the modeling of sequential dependence between items shown in Fig. 1(a).

To this end, we propose a novel method (named HyperS^{2}Rec) which simultaneously considers both consistency and sequential dependence between items. Specifically, it first learns item representations with hypergraph structure and sequence structure, respectively. Then the final item representations are generated via an attention mechanism. Moreover, the reversed position embedding mechanism and soft attention mechanism are exploited to integrate the impact of position and obtain session representation. Finally, the model outputs the probabilities that each candidate item becomes the next click. The main contributions of this paper are summarized as follows.

- We propose an efficient session-based recommendation method named HyperS^{2}Rec. It is capable of capturing both consistency and sequential dependence between items that are widely available in real-world scenarios.
- It learns two kinds of item representations with hypergraph convolutional network (HGCN) and gated recurrent unit (GRU), respectively. Moreover, it flexibly gathers relatively important information from the learned representations for recommendation with an attention mechanism.
- We conduct extensive experiments on Tmall, RetailRocket and Diginetica datasets. The experimental results show that the proposed HyperS^{2}Rec outperforms the state-of-the-art methods.

2. Related work

In this section, we review session-based recommendation methods related to this work from three aspects: traditional methods, deep learning-based methods and GNN-based methods.

2.1. Traditional methods

Since user profiles are often not available in some scenarios, it is natural to leverage the relationships between items to generate results. Item-KNN (Sarwar, Karypis, Konstan, & Riedl, 2001) recommends K items similar to previous clicks based on cosine similarity. However, it does not consider the effects of interactive order. Inspired by word embedding methods, CoFactor (Liang, Altosaar, Charlin, & Blei, 2016) decomposes interaction matrix and co-occurrence matrix to improve recommending performance. The information available in session-based recommendation is only the short-term historical interactions of anonymous users, thus simple matrix factorization is not well-suited. PRME-G (Feng et al., 2015) adopts metric embedding to overcome the shortcoming of matrix factorization. PME (Wu et al., 2013) utilizes distances between items and users to indicate their relationships and makes recommending results based on the last given item. FPMC (Rendle, Freudenthaler, & Schmidt-Thieme, 2010) fuses personalized first-order Markov chains and a matrix factorization model to learn transition matrix for each user. Nevertheless, it only focuses on the transition between two adjacent items in a sequence, which has difficulty in capturing complex high-order sequence patterns.

2.2. Deep learning-based methods

As RNN has shown advantages in modeling sequential data, it is extensively employed in the field of session-based recommendation. GRU4Rec (Hidasi, Karatzoglou et al., 2016) attempts to employ RNN to recommender systems for the first time, and regards the clicks of a user as a sequence. Besides, it models sessions with the improved GRU layer. NARM (Li et al., 2017) simulates the sequential behavior of users and captures the main purpose of a session via a hybrid encoder. STAMP (Liu et al., 2018) is designed based on multilayer perceptron and attention network. It takes both of the short-term and long-term interest of a session into account. Hierarchical RNN (Quadrana, Karatzoglou, Hidasi, & Cremonesi, 2017) extends existing RNN modeling via integrating one GRU level. It is able to capture dynamic preferences by transferring with cross-session information. DREAM (Yu, Liu, Wu, Wang, & Tan, 2016) is designed to learn dynamic interests and the global sequential information it obtained reflects the interactions of a user over time. ReLaVaR (Chatzis, Christodoulou, & Andreou, 2017) cooperates RNN with variational inference to alleviate data sparsity. p-RNN (Hidasi, Quadrana, Karatzoglou & Tikk, 2016) utilizes parallel RNN to model clicking items and their features in a session and exploits a novel training strategy to improve recommending performance. Jannach and Ludewig fused the advantages of RNN and heuristics-based KNN to effectively use sequential signals and co-occurrence information in a session (Jannach & Ludewig, 2017).

Apart from RNN-based methods, there are still lots of deep learning-based methods for session-based recommendation. HMN (Song, Cao, Zhang, & Xu, 2019) learns memory matrices from two aspects: item-level and feature-level. In addition, it adopts hierarchical attention mechanism and selects multi-level features to obtain session representation. CSRM (Wang et al., 2019) utilizes two memory modules to consider the information of the ongoing session and the cooperative neighbor information in parallel, respectively. Caser (Tang & Wang, 2018) investigates the effect of recent items on recommending and learns sequential patterns via convolution operator. Tuan and Phuong proposed to utilize content features as the supplementary information of clicking items (Tuan & Phuong, 2017). They adopted 3-dimensional convolutional neural network to model different types of input data. HierTCN (You et al., 2019) is comprised of high-level and low-level model. It learns long-term interests and short-term interactions via RNN and temporal convolutional network, respectively. DIDN (Zhang, Lin et al., 2022) combines item, user and temporal information to capture the dynamic intention of a session. Besides, it filters out the noise for sessions via a well-designed denoising module.

Although deep learning-based methods have made great success, the shortcoming of them is obvious. They excessively rely on sequential dependencies between adjacent items, which ignores transitional information between non-adjacent items.

2.3. GNN-based methods

SR-GNN (Wu, Tang et al., 2019) is the first work to model sessions as graph. It adopts GGNN to capture complex item transitions, and fuses the local and global features of a session through an attention mechanism. GC-SAN (Xu et al., 2019) exploits GNN to capture local dependencies and then learns long-range dependencies with the help of self-attention mechanism. TA-GNN (Yu et al., 2020) uses a target-aware attention to activate the interests of users. It learns different interest representations for different target items. FGNN (Qiu, Li, Huang, & Yin, 2019) regards a weighted graph attention network as the encoder of item features. Moreover, it applies a readout function to generate session representations. MSGIFSR (Guo et al., 2022) learns interactions between consecutive units from a high-level perspective via multi-granularity session graph. It simultaneously takes order-variant and order-invariant relations into account and achieves promising performance. However, the methods above merely exploit the directed session graph to model the item transitions in the ongoing session, which neglects the rich information hidden across sessions.

To tackle this problem, many efforts have been made recently. A-PGNN (Zhang et al., 2020) learns structural information in the user behavior graph via a personalized GNN. Besides, the information of historical sessions is integrated into the current session to achieve personalized recommendation. GAG (Qiu, Yin, Huang, & Chen, 2020) constructs session graph based on the ongoing session. The highlight of this work is that it takes the global attributes into consideration to learn more comprehensive representations. SGNN-HN (Pan, Cai, Chen, Chen, & de Rijke, 2020) introduces virtual nodes as star nodes to construct star graph. It uses star GNN to learn complex transitions between items that are not directly connected and alleviates over-fitting via highway networks. GCE-GNN (Wang et al., 2020) incorporates relevant information in other sessions into the current session. Moreover, it performs graph attention network on session graph and global graph to represent more complicated item transitions. DGTN (Zheng, Liu, Li, & Wu, 2020) integrates the current session and neighbor sessions into the same graph to capture similar behavior patterns. GNN-GNF (Feng, Cai, Wei, & Li, 2022) first filters out the noise within a session by an item-level denoising module. Then, it selects related sessions based on the intention of the current session via edge matching to construct global graph.

Although GNN has achieved promising performance, the above methods are proposed mainly based on a simple graph. Thus, they are not well-suitable for capturing high-order relationships between items (Gao et al., 2021), which limits their representative ability. Therefore, GNN is extended to hypergraph for modeling complex high-order transitions. SHARE (Wang et al., 2021) constructs a hypergraph for each session, and uses a contextual sliding window to model the correlation of items. Besides, it utilizes HGAT to distinguish items with different degrees of importance. DHCN (Xia et al., 2021) is a recent method which is closely related to our work. It performs graph convolution on hypergraph and line graph to obtain item-level and session-level representations, with self-supervised learning for training. Nevertheless, it ignores the sequential dependence of short-term session data in real world. Different from DHCN, the HyperS^{2}Rec proposed in this paper fully considers both the consistency and sequential dependence between items. The integration of hypergraph-structure and sequential information enables the model to dig out the real intentions of the current session to a great extent.

Table 1
The notations used in this paper.

Notations	Descriptions
$\mathcal{G}$	The constructed hypergraph according to sessions.
H	The incidence matrix of hypergraph $\mathcal{G}$.
D, B	The degree matrix of vertex/hyperedge.
$\lvert S\rvert$, $\lvert V\rvert$	The number of sessions/items.
N_i	The length of s_i.
c_i	The initial representation of item v_i.
h_i	The representation of item v_i based on HGCN.
g_i	The representation of item v_i based on sequential information embeddings.
α_h	The weight coefficients of item representations based on HGCN.
α_g	The weight coefficients of item representations based on sequential information embeddings.
$f_{i,t}$	The final representation of tth item in s_i.
$\beta_{i,t}$	The weight coefficient of tth item in s_i.
u_i	The final representation of s_i.

3. Preliminaries

Definition 1 (*Hypergraph*). A hypergraph (Bai, Zhang, & Torr, 2021) is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of N vertices and $\mathcal{E}$ is a set of M hyperedges. Each hyperedge $e_j \in \mathcal{E}$ is assigned a positive weight W_{jj}, and all the weights formulate a diagonal matrix $W \in \mathbb{R}^{M\times M}$. The incidence matrix of a hypergraph $\mathcal{G}$ can be denoted as $H \in \mathbb{R}^{N\times M}$. If a vertex $v_i \in \mathcal{V}$ is connected by a hyperedge $e_j \in \mathcal{E}$, $H_{ij} = 1$, otherwise 0. For each vertex and hyperedge, their degree D_{ii} and B_{jj} are defined as $D_{ii} = \sum_{j=1}^{M} W_{jj} H_{ij}$; $B_{jj} = \sum_{i=1}^{N} H_{ij}$ respectively. $D \in \mathbb{R}^{N\times N}$ and $B \in \mathbb{R}^{M\times M}$ are diagonal matrices.

Problem (*Session-based Recommendation*). Let $S = \{s_1, s_2, \dots, s_{|S|}\}$ denote a set of sessions. Let $V = \{v_1, v_2, \dots, v_{|V|}\}$ denote a set consisting of all unique items involved in all sessions. $|S|$ and $|V|$ are the number of sessions and items, respectively. Each anonymous session can be represented by a list $s_i = \left[v_{i,1}, v_{i,2}, v_{i,3}, \dots, v_{i,N_i}\right]$ ordered by timestamps, where N_i denotes the length of s_i and $v_{i,t} \in V (1 \le t \le N_i)$ represents the tth item in s_i. The task of session-based recommendation is to predict the next item for s_i, i.e., v_{i,N_i+1}. Formally, it outputs the probabilities $\hat{y}$ for all candidate items, where $\hat{y}_i$ represents the probability of v_i being the next item. The top-K items among all candidates are selected for recommendation according to $\hat{y}$. The notations mainly used in this paper are summarized in Table 1.

4. The proposed method

4.1. Overall framework

The overall framework of the proposed HyperS^{2}Rec is shown in Fig. 2. It is comprised of four components: (1) Hypergraph information

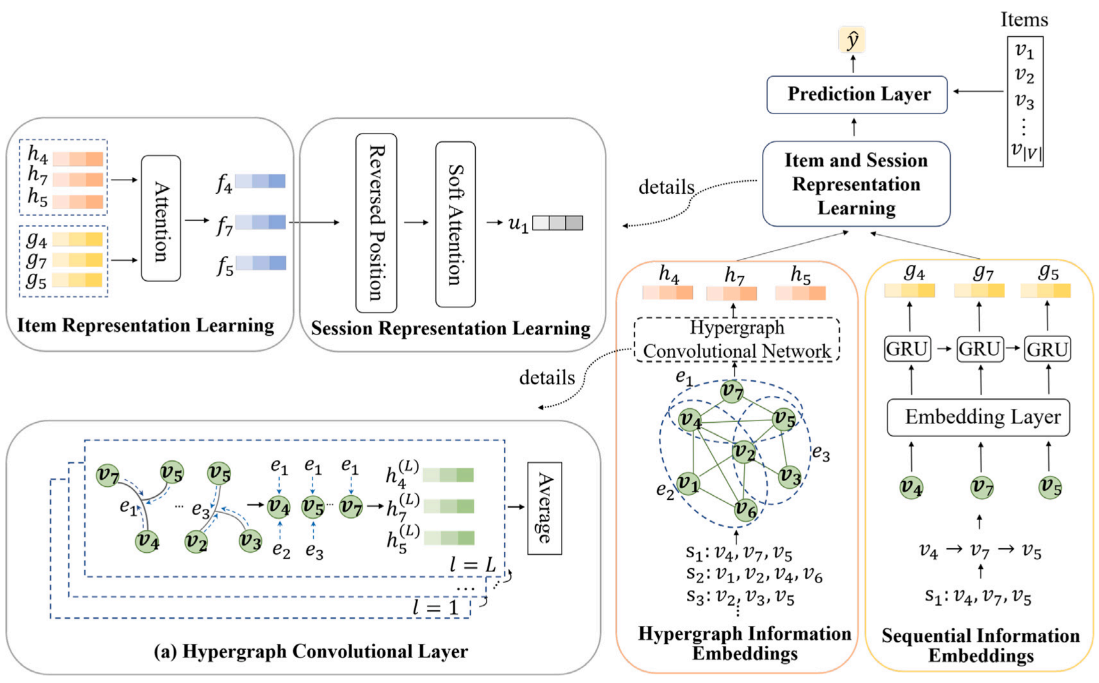
Items
Prediction Layer
Item and Session Representation Learning
details
Attention
Reversed Position
Soft Attention
Item Representation Learning
Session Representation Learning
Hypergraph Convolutional Network
GRU
Embedding Layer
Average
(a) Hypergraph Convolutional Layer
Hypergraph Information Embeddings
Sequential Information Embeddings

Fig. 2. The overall framework of HyperS^{2}Rec. First, we model sessions as a hypergraph and obtain item representations based on HGCN. Meanwhile, we utilize GRU to learn sequential information in s_i. We then generate the final item representations via the attention mechanism. Moreover, the reversed position embedding mechanism and soft attention mechanism are utilized to obtain session representation u_i. Finally, the prediction layer outputs the probabilities of candidates being the next item in s_i.

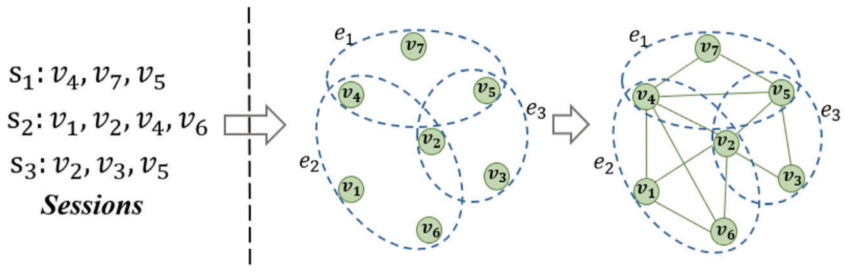

Fig. 3. The process of hypergraph construction.

embeddings. We model sessions as a hypergraph and obtain item representations based on HGCN; (2) Sequential information embeddings. We employ GRU to get item representations based on sequential information embeddings; (3) Item and session representation learning. First, the final item representations are learned via an attention mechanism. Then, they are fused to obtain session representation via the reversed position embedding mechanism and the soft attention mechanism; (4) Prediction layer. The model calculates the probabilities that each candidate item becomes the next click, and then selects the top-K items for recommendation.

4.2. Hypergraph information embeddings

Hypergraph Construction. Inspired by Xia et al. (2021), we model sessions as an undirected hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each session is modeled as a hyperedge and each item is modeled as a vertex. Fig. 3 shows the process of constructing a hypergraph from a series of sessions. First, three sessions s_1, s_2, s_3 are regarded as three hyperedges e_1, e_2, e_3, respectively. Then the items in each session are connected in pairs regardless of the order. This way is helpful for the HyperS^{2}Rec to capture the consistency between items in a session.

Hypergraph Convolutional Network. It generalizes graph convolutional network (GCN) to hypergraph, and takes the hyperedges as transition during information propagation. The procedures are divided into the following two phases: (1) information aggregation from vertices to hyperedges, and (2) information aggregation from hyperedges to vertices. Specifically, the information of each vertex is aggregated into the hyperedge it located. Thus, the representation of each hyperedge is obtained. After that, look up the hyperedges connected to each vertex, and the information of these hyperedges is aggregated into the vertices. As a result, the representation of each vertex is generated.

Fig. 2(a) shows the details of the hypergraph convolutional layer. First of all, the information of vertex v_4, v_5, v_7 is aggregated into the hyperedge e_1. The information of vertex v_1, v_2, v_4, v_6 is aggregated into the hyperedge e_2. The information of vertex v_2, v_3, v_5 is aggregated into the hyperedge e_3. Then, as vertex v_4 appears in both hyperedge e_1 and e_2, the information of e_1, e_2 is aggregated into v_4. Similarly, the information of hyperedge e_1, e_3 is aggregated into vertex v_5. In this way, the representations of all vertices are obtained. It is worth noting that, the HGCN enables HyperS^2 Rec to capture the information of closely related sessions with the current one, which further improves the learning of item representations.

Similar to the convolutional network defined on simple graph, the challenge of HGCN is how to make the representations propagate between convolutional layers. Wu, Souza et al. (2019) verified that removing the nonlinear activation function and weight matrix between layers in the process of propagation can not only reduce the additional complexity, but also have no negative impact on downstream tasks.

Thus, following previous work (Bai et al., 2021; Wu, Souza et al., 2019; Xia et al., 2021), we define HGCN as follows:

$$x_t^{(l+1)} = \sum_{i=1}^{N} \sum_{j=1}^{M} H_{tj} H_{ij} W_{jj} x_i^{(l)}, \tag{1}$$

where $x_t^{(l+1)}$ denotes the representation of the tth node at the $(l+1)$th layer. W_{jj} is set to 1 for each j.

The matrix of Eq. (1) with row normalization is:

$$X^{(l+1)} = D^{-1} H W B^{-1} H^T X^{(l)}, \tag{2}$$

where $W \in \mathbb{R}^{|S|\times|S|}$ denotes the weight matrix of hyperedges. Here it is an identity matrix. $D \in \mathbb{R}^{|V|\times|V|}$ and $B \in \mathbb{R}^{|S|\times|S|}$ are the degree matrices of vertex and hyperedge, respectively. $H \in \mathbb{R}^{|V|\times|S|}$ is the incidence matrix of the constructed hypergraph. $X^{(l)}, X^{(l+1)} \in \mathbb{R}^{|V|\times d}$ denotes the representations of all items at the lth layer and $(l+1)$th layer, respectively. d is the representational dimension. The input of 0th layer is the initialized item representations $C = \left[c_1, c_2, \dots, c_{|V|}\right]$ which are obtained according to item id, i.e., $X^{(0)} = C$.

We pass $X^{(0)}$ through L hypergraph convolutional layers, and average item representations learned by each layer to obtain item representations based on HGCN:

$$X_h = \frac{1}{L+1} \sum_{l=0}^{L} X^{(l)}. \tag{3}$$

4.3. Sequential information embeddings

It has been demonstrated that RNN is effective in modeling sequences. As a variant of RNN, GRU is able to well alleviate the problem of long-range dependencies. Besides, it has fewer parameters and faster training speed than Long-Short Term Memory (LSTM). Therefore, we design a module named sequential information embeddings based on GRU to effectively capture the sequential dependence between items in a session.

The initial hidden state g_i is set to zero vector. For the initial representation c_i of item v_i, GRU updates the hidden states g_i as follows:

$$r_i = \sigma\left(W_r c_i + U_r g_{i-1}\right), \tag{4}$$

$$a_i = \sigma\left(W_a c_i + U_a g_{i-1}\right), \tag{5}$$

$$\tilde{g}_i = \tanh\left(W_{\tilde{g}} c_i + U_{\tilde{g}}\left(r_i \odot g_{i-1}\right)\right), \tag{6}$$

$$g_i = \left(1 - a_i\right) \odot g_{i-1} + a_i \odot \tilde{g}_i, \tag{7}$$

where $W_r, W_a, W_{\tilde{g}} \in \mathbb{R}^{d\times d}, U_r, U_a, U_{\tilde{g}} \in \mathbb{R}^{d\times d}$ are the learnable parameters. $\sigma(\cdot)$ denotes sigmoid function. $\odot$ denotes element-wise multiplication operator. The reset gate r_i determines how to combine the information of the current moment with the previous. The update gate a_i controls the proportion of the memory from the previous moment to the current. $\tilde{g}_i$ remembers the state of the current moment. We can obtain the item representations based on sequential information embeddings X_g by employing GRU.

4.4. Item and session representation learning

The attention mechanism has the ability to automatically collect the relatively important information needed by assigning diverse weights. The weights it generates can be adjusted continuously to select important information under different circumstances. Based on the above advantages, we use an attention mechanism to learn the final item representations and a soft-attention mechanism to obtain session representations, respectively.

Item representation learning. For item v_i, we exploit an attention mechanism to adaptively calculate the relative importance between the above two aspects:

$$\mu_h^i = {q_1}^T \tanh\left(W_1 h_i + b_1\right), \tag{8}$$

$$\mu_g^i = {q_1}^T \tanh\left(W_1 g_i + b_1\right), \tag{9}$$

where h_i and g_i denote the representation of item v_i based on HGCN and sequential information embeddings, respectively. $q_1 \in \mathbb{R}^d, W_1 \in \mathbb{R}^{d\times d}$ and $b_1 \in \mathbb{R}^d$ are learnable attention vector, weight matrix and bias vector, respectively.

The softmax function is employed to convert the importance into weight coefficients:

$$\alpha_h^i = \frac{\exp\left(\mu_h^i\right)}{\exp\left(\mu_h^i\right) + \exp\left(\mu_g^i\right)} = 1 - \alpha_g^i, \tag{10}$$

where α_h^i and α_g^i are the weight coefficients of item representations based on HGCN and sequential information embeddings, respectively.

The final representation of item v_i can be obtained with the following formula:

$$f_i = \alpha_h^i h_i + \alpha_g^i g_i. \tag{11}$$

As a result, the representations of all the items in s_i are learned, i.e., $F_i = \left[f_{i,1}, f_{i,2}, \dots, f_{i,N_i}\right]$.

Session representation learning. It has been verified that each item carries the positional information (Qiu, Huang, Chen, & Yin, 2021) and the reversed position is playing more important role in exploring users' intention (Wang et al., 2020). Therefore, in this paper, the reversed position that each item carries is taken into account for session representation learning. The position embedding matrix is denoted as $P = \left[p_1, p_2, \dots, p_{N_i}\right]$, where $p_i \in \mathbb{R}^d$ refers to a position vector for position i. The position information is integrated with the following formula:

$$f_{i,t}^* = \tanh\left(W_2\left[f_{i,t} \| p_{N_i-t+1}\right] + b_2\right), \tag{12}$$

where $\|$ is concatenation operator. N_i is the length of s_i. $W_2 \in \mathbb{R}^{d\times 2d}$ and $b_2 \in \mathbb{R}^d$ are learnable parameters, respectively.

The soft attention mechanism (Wu, Tang et al., 2019) is adopted to distinguish different contributions of items to the final session representations. First, the representation of s_i is obtained by averaging its item representations.

$$s_i' = \frac{1}{N_i} \sum_{t=1}^{N_i} f_{i,t}. \tag{13}$$

Then, it learns the weight coefficient for each item. The formula is shown as follows:

$$\beta_{i,t} = {q_2}^T \sigma\left(W_3 s_i' + W_4 f_{i,t}^* + b_3\right), \tag{14}$$

where $\sigma(\cdot)$ denotes sigmoid function. $q_2 \in \mathbb{R}^d$ is an attention parameter. $W_3, W_4 \in \mathbb{R}^{d\times d}$ and $b_3 \in \mathbb{R}^d$ are learnable weight matrix and bias vector, respectively. The larger the $\beta_{i,t}$, the more important the tth item is to capturing the intention of s_i.

Finally, the session representation u_i is calculated as follows.

$$u_i = \sum_{t=1}^{N_i} \beta_{i,t} f_{i,t}. \tag{15}$$

4.5. Prediction layer

Given an item $v_j \in V$, the score of v_j in session s_i is defined as the similarity between the representations u_i and f_j, shown in Eq. (16). Thus, the score vector $\hat{z}_{s_i}$ can be obtained with $\hat{z}_{s_i,j}$ being the jth element.

$$\hat{z}_{s_i,j} = {u_i}^T f_j, \tag{16}$$

Then, we adopt the softmax function to get the prediction vector $\hat{y}_{s_i}$, which is written as follows:

$$\hat{y}_{s_i} = \mathrm{softmax}(\hat{z}_{s_i}), \tag{17}$$

where $\hat{y}_{s_i} \in \mathbb{R}^{|V|\times 1}$, and $\hat{y}_{s_i,j}$ denotes the probability of v_j becoming the next item in session s_i. The top-K items among all candidates are selected for recommendation according to $\hat{y}_{s_i}$.

For session s_i, the loss function is defined as follows:

$$\mathcal{L}(\hat{y}_{s_i}) = -\sum_{j=1}^{|V|} y_{s_i,j} \log\left(\hat{y}_{s_i,j}\right) + \left(1 - y_{s_i,j}\right) \log\left(1 - \hat{y}_{s_i,j}\right), \tag{18}$$

where $y_{s_i,j}$ is the ground truth to denote whether v_j is the next-clicked item in session s_i.

4.6. Algorithm and complexity analysis

Algorithm 1 presents the training process of HyperS^{2}Rec. For modeling session $s_i = \left[v_{i,1}, v_{i,2}, v_{i,3}, \dots, v_{i,N_i}\right]$, the time complexity mainly includes three parts: $O\left(N_i d L + N_i d^2 + |V|d\right)$ from item representation learning, $O\left(N_i d^2\right)$ from session representation learning and $O(|V|d)$ from model prediction and optimization. Therefore, the total time complexity of HyperS^{2}Rec is $O\left(T|S|\left(N_i d L + N_i d^2 + |V|d\right)\right)$, where $|V|$ denotes the number of unique items, d denotes the representational dimension, L is the number of HGCN layers, N_i is the length of session s_i and $|S|$ is the number of sessions. The space complexity of HyperS^{2}Rec is $O\left(\left(d + d^2 + pd + |V|d\right)\right.$ in total, where p is the maximum length of sessions.

Algorithm 1 Training process of HyperS^{2}Rec

```
Input: session set $S = \{s_1, s_2, \dots s_{|S|}\}$, the number of HGCN layer $L$, maximum
    epoch $T$
Output: Trainable parameters $\Phi$
 1: Initialize parameters $\Phi$;
 2: repeat
 3:   for $t$ in range $[0, T]$ do
 4:     for $s_i$ in $S$ do
 5:       for $l$ in range $[1, L]$ do
 6:         Learn $l$th item representations according to Eq. (2);
 7:       end for
 8:       Learn item representations based on HGCN according to Eq. (3);
 9:       Learn item representations based on sequential information
          embeddings according to Eq. (4)–(7);
10:       Obtain final representation of each item in $s_i$ according to
          Eq. (8)–(11);
11:       Learn the representation of $s_i$ according to Eq. (12)–(15);
12:       Predict the probabilities of all candidates being the next item in $s_i$
          according to Eq. (16)–(17);
13:     end for
14:     Update $\Phi$ with gradient descent;
15:   end for
16: until Eq. (18) converges;
17: Return $\Phi$
```

5. Experiments

5.1. Datasets

We carried out experiments on the following three datasets to evaluate the performance of HyperS^{2}Rec.

Tmall[1]: It comes from the IJCAI-15 competition. It comprises the shopping histories of anonymous users on the "Double 11" day and the previous 6 months on Tmall online shopping website.

RetailRocket[2]: It is a Kaggle competition dataset published by an e-commerce company. It includes browsing statistics of anonymous users over a period of 4.5 months.

[1] https://tianchi.aliyun.com/dataset/dataDetail?dataId=42

[2] https://www.kaggle.com/retailrocket/ecommerce-dataset

Table 2
Statistics of datasets.

	Tmall	RetailRocket	Diginetica
# of training sessions	351,268	433,648	719,470
# of test sessions	25,898	15,132	60,858
# of clicks	818,479	710,586	982,961
# of items	40,728	36,968	43,097
Average length	6.69	5.43	5.12

Diginetica[3]: It comes from CIKM Cup 2016 and includes information of sessions extracted from e-commerce search engine logs. Only transactional data is utilized in our experiments.

The testing sets consist of the sessions of the last week, and the training sets are the rest historical sessions. Following the previous work (Li et al., 2017; Liu et al., 2018; Wu, Tang et al., 2019), all sessions that the length is 1 and all items that appear less than 5 times were filtered out. In addition, we adopted splitting operation for each session. That means, for each session $s_i = \left[v_{i,1}, v_{i,2}, v_{i,3}, \dots, v_{i,N_i}\right]$, a series of sequences and corresponding labels $\left(\left[v_{i,1}\right], v_{i,2}\right), \left(\left[v_{i,1}, v_{i,2}\right], v_{i,3}\right), \dots, \left(\left[v_{i,1}, v_{i,2}, \dots, v_{i,N_i-1}\right], v_{i,N_i}\right)$ are generated, where the label of each sequence is its last item. The statistics of three datasets are described in Table 2.

5.2. Baselines

We compared the proposed HyperS^{2}Rec with the following eleven competitive baselines:

Item-KNN (Sarwar et al., 2001): The K items most similar to previous clicks are recommended based on cosine similarity.

FPMC (Rendle et al., 2010): The personalized first-order Markov chain and matrix factorization are combined in this method to accomplish sequential recommendation.

GRU4Rec (Hidasi, Karatzoglou et al., 2016): It regards the clicks of a user as a sequence, and models the interactive sequences via the improved GRU layer.

STAMP (Liu et al., 2018): It is designed based on multilayer perceptron and attention network. Besides, it proposes to capture the short-term interest through the most recently clicked item.

CoSAN (Luo et al., 2020): It constructs the dynamic item representations and neighbor session representations. Meanwhile, it models the long-range dependencies between collaborative items to predict the user's intention.

DIDN (Zhang, Lin et al., 2022): It combines item, user, and temporal information to learn dynamic intention. Besides, a denoising module is devised to filter out the noise for sessions.

SR-GNN (Wu, Tang et al., 2019): It models each session as a subgraph. The item representations are learned by GGNN, and the session representations are obtained by fusing local and global features through an attention network.

GC-SAN (Xu et al., 2019): It utilizes both GGNN and self-attention mechanism to enrich contextualized representations of items.

GNN-GNF (Feng et al., 2022): It constructs global graph after a data preprocessing module which includes item-level and session-level filter module.

SHARE (Wang et al., 2021): A contextual sliding window is designed to model dynamic user interests, and hypergraph convolutional attention network is integrated to capture preferences of users.

DHCN (Xia et al., 2021): To obtain item-level and session-level representations, the graph convolution is performed on hypergraph and line graph. Besides, self-supervised is applied into the training of the network.

[3] http://cikm2016.cs.iupui.edu/cikm-cup/

Table 3
The performance of the proposed HyperS^{2}Rec and competitors over three datasets. The best and second best results are highlighted in bold and underline.

Methods	Tmall				RetailRocket				Diginetica			
	H@10	M@10	H@20	M@20	H@10	M@10	H@20	M@20	H@10	M@10	H@20	M@20
Item-KNN	6.65	3.11	9.15	3.31	22.48	10.43	28.97	10.98	25.07	10.77	35.75	11.57
FPMC	13.1	7.12	16.06	7.32	25.99	13.38	32.37	13.82	15.43	15.43	26.53	6.95
GRU4Rec	9.47	5.78	10.93	5.89	38.35	23.27	44.01	23.67	17.93	17.93	29.45	8.33
STAMP	22.63	13.12	26.47	13.36	42.95	24.61	50.96	25.17	33.98	14.26	45.64	14.32
CoSAN	24.16	11.78	28.39	12.52	43.81	23.8	52.47	24.4	34.75	14.29	48.34	15.22
DIDN	24.38	13.89	29.56	13.96	43.51	26.93	51.4	27.16	40.11	17.41	53.44	18.52
SR-GNN	23.41	13.45	27.57	13.72	43.21	26.07	50.32	26.57	36.86	15.52	50.73	17.59
GC-SAN	15.75	8.64	19.28	8.89	44.1	26.92	51.18	27.4	37.86	16.89	50.84	17.79
GNN-GNF	24.96	13.88	28.93	14.22	44.87	26.81	52.79	27.21	37.67	15.76	51.61	17.77
SHARE	25.04	13.98	29.69	14.24	46.47	26.59	54.16	27.11	39.52	17.12	52.73	18.05
DHCN	25.16	13.92	30.46	14.28	47.93	28.28	55.75	28.82	39.78	17.44	53.09	18.35
HyperS^{2}Rec	**27.26**	**14.98**	**32.91**	**15.39**	**49.11**	**29.44**	**56.71**	**29.95**	**40.52**	**18.09**	**54.13**	**18.91**
Improv(%)	8.35	7.15	8.04	7.77	2.46	4.1	1.72	3.92	1.02	3.7	1.29	2.1

5.3. Evaluation metrics

Two ranking based metrics are adopted to evaluate the recommending results.

HR@K(Hit Rate): HR@K is used to measure the accuracy of the recommendation. It refers to the proportion of correct items in the generated K recommended items and is defined as:

$$HR@K = \frac{Hit_n}{N}, \tag{19}$$

where Hit_n denotes the number of correct items included in the generated K ranked list. N is the total number of the test set. The larger the HR@K, the more accurate the recommendation result is.

MRR@K(Mean Reciprocal Rank): MRR@K emphasizes the rank of the correct item. It is used to measure the order of recommendation ranking and is defined as:

$$MRR@K = \frac{1}{N}\sum_{i=0}^{N}\frac{1}{\text{rank}(i)}, \tag{20}$$

where N is the total number of the test set. rank(i) denotes the rank of the correct item in the generated list in the ith test sample. If the correct item is not in the generated list, then $1/(\text{rank}(i))$ is set to 0. A larger MRR@K indicates that the rank of the correct item is higher.

5.4. Parameter settings

We use Gaussian distribution with a mean of 0 and a standard deviation of 0.1 to initialize parameters. For the sake of fairness, the representational dimension and batch size in all methods are set to 100, and the epoch is 30. We use Adam to optimize our model. The learning rate is selected from {0.0001, 0.0005, 0.001, 0.005, 0.01}, and will decay by 0.1 after 3 epoch. The optimal number of hypergraph convolutional layers on three datasets is searched in {1, 2, 3, 4, 5}, respectively. Moreover, for all the baselines, we adopt the best parameter settings as they reported.

5.5. Experimental results and analyses

The experimental results are shown in Table 3. As demonstrated in Table 3, HyperS^{2}Rec outperforms all the baselines on three datasets. Specifically, we have the following interesting observations and analyses.

(1) Item-KNN performs poorly for the reason that it only models the similarity between items, which ignores the interactive order in a session. Thus, it is incapable of capturing the transitions between items. In contrast to Item-KNN, FPMC demonstrates its effectiveness. It combines matrix factorization with Markov chains. This explains, to some extent, the importance of transitions between items for capturing user preferences.

(2) GRU4Rec is the first attempt of RNN in the field of session-based recommendation. Although its performance is not as good as FPMC on Tmall dataset, RNN-based methods significantly outperform traditional methods in general, indicating that RNN is well suitable for modeling sequential data. STAMP uses recurrent units to encode the interactive sequences, and assigns weights to items in a session through an attention mechanism. Experimental results show that STAMP outperforms GRU4Rec, indicating that assigning different weights to different items is particularly essential to accurately capture the intention of a session. Moreover, STAMP treats the last clicked item as short-term interest, demonstrating the significance of capturing short-term memory for predicting the next interaction. Nevertheless, the gradient vanishing and exploding problem existed in RNN make the training process of the above methods become tricky. CoSAN integrates the information of neighbor sessions for the current session by utilizing collaborative self-attention network. As shown in Table 3, the improvement on HR is promising but disappointing on MRR. This confirms that incorporating the information of related sessions can indeed enrich the representation of the current session, but it also brings some noise. Diginetica dataset contains more items and clicks compared with the other two datasets, which may bring more noises. Thus, DIDN achieves competitive performance on Diginetica dataset compared with other baselines. The experimental results of DIDN demonstrate that noisy clicks indeed affect the prediction of intentions, and this problem can be alleviated by effective denoising methods. Besides, it is also essential to pay attention to the dynamic changes of intention over time.

(3) The experimental results show that GNN-based methods are more competitive. SR-GNN models sessions as graph structure, and learns item representations through GGNN. GC-SAN introduces self-attention to learn long-range dependencies in a session which is not able to be learned by GNN. However, the performance of GC-SAN on Tmall dataset is disappointing. The possible reason is that the average length of sessions in Tmall dataset is relatively long, and the long-range dependencies between items are not as strongly as assumed in GC-SAN. It also further verifies our hypothesis that the items have not only sequential dependence, but also consistency. Experimental results show that GNN-GNF outperforms SR-GNN and GC-SAN, demonstrating that denoising can help capture the intention more accurately. SHARE and DHCN utilize hypergraph to model sessions and gain great improvements, which indicates the superiority of hypergraph modeling.

(4) The proposed HyperS^{2}Rec performs the best on three datasets. Compared with SR-GNN and GC-SAN which model sessions as simple graph structure, HyperS^{2}Rec models them as hypergraph, and connects any two items in a session. The characteristic of hyperdege enables HyperS^{2}Rec to capture consistency between items. Compared with SHARE and DHCN, HyperS^{2}Rec considers the sequential dependence between items within a session, which enables it to capture more complicated item transitions. In summary, the experimental results show that HyperS^{2}Rec is able to consider both hypergraph-structured

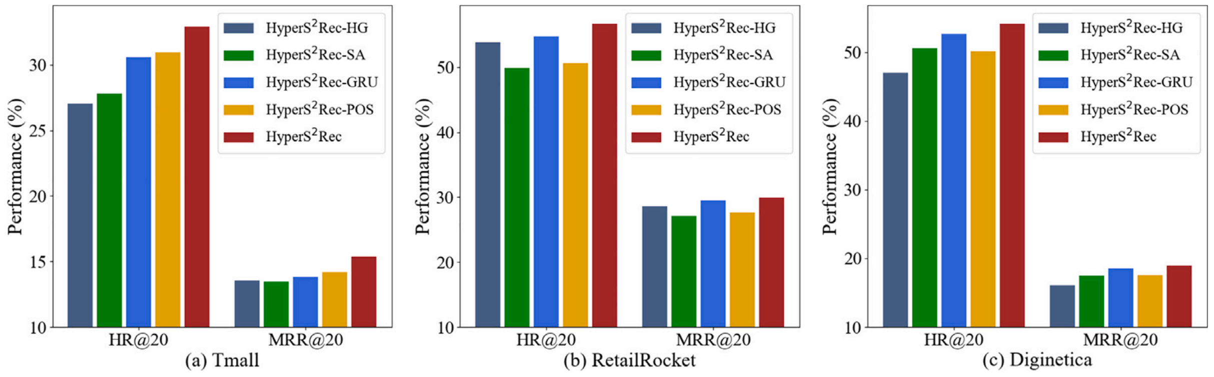

Fig. 4. The performance of HyperS^{2}Rec and its variants on three datasets.

information and sequential information simultaneously. Thus it has more powerful representational ability to capture the real intention of a session.

5.6. Ablation study

We carried out ablation experiments to further study the impact of each component in HyperS^{2}Rec. We defined the following four variants:

- HyperS^{2}Rec-HG: This model removes the hypergraph information embeddings from the proposed HyperS^{2}Rec. It considers the representations obtained by GRU as the final item representations. This variant is designed to investigate the effectiveness of hypergraph-structured information.
- HyperS^{2}Rec-GRU: This model removes the sequential information embeddings from the proposed HyperS^{2}Rec. It takes the representations obtained by HGCN as the final item representations. This variant is designed to investigate the effectiveness of sequential information.
- HyperS^{2}Rec-POS: This is a variant which removes the reversed position embedding mechanism from the proposed HyperS^{2}Rec. It is designed to investigate the effects of reversed positional information.
- HyperS^{2}Rec-SA: It replaces the session representation described in Eq. (15) by averaging item representations described in Eq. (13). This variant is designed to explore the impact of soft attention mechanism.

The experimental results of HyperS^{2}Rec and four variants are illustrated in Fig. 4. Specifically, we have the following conclusions:

(1) HyperS^{2}Rec outperforms HyperS^{2}Rec-HG and HyperS^{2}Rec-GRU on three datasets. It indicates that either sequential information or hypergraph-structured information is insufficient to predict the next-item. Besides, the performance of HyperS^{2}Rec-HG decreases more than HyperS^{2}Rec-GRU, which implies that the consistency between items is more important than sequential dependence. In summary, the hidden hypergraph-structured information captured by HGCN and sequential information obtained by GRU both contribute to the final session representations.

(2) HyperS^{2}Rec achieves a better performance than HyperS^{2}Rec-POS. It shows us the effectiveness of the reversed position embedding mechanism. In other words, the reversed positional information of items is playing a very important role in recommendation. Therefore, the reversed position embedding mechanism in HyperS^{2}Rec is essential to reflect the real intention of users.

(3) HyperS^{2}Rec outperforms HyperS^{2}Rec-SA, which indicates that items in a session indeed contribute differently to capturing the preference of the current session. Therefore, the performance of HyperS^{2}Rec benefits from the soft attention mechanism which assigning different weights to different items when generating session representations.

5.7. Parameter analysis

Impact of HGCN layers. In order to explore the effects of the number of HGCN layers, we ranged the number of layers within {1, 2, 3, 4, 5}. Fig. 5 shows how the change of convolutional layers affect the performance of HyperS^{2}Rec. As demonstrated in Fig. 5, the best result is achieved when the number of layers is set to three on RetailRocket dataset and Diginetica dataset. Nevertheless, one layer is the best on Tmall dataset, and the performance decreases when the number of layers increases.

In addition to learning consistency between items in a session, HGCN in HyperS^{2}Rec is able to capture the information of neighbors' sessions. For RetailRocket dataset and Diginetica dataset, the average length of sessions is relatively short. So it is necessary to supplement the current session with information of related sessions. As shown in Fig. 5, the performance of HyperS^{2}Rec is not the best when the number of layers is set to one and two. Meanwhile, the performance decreases when the number of layers is greater than three. The possible reason is over-smoothing. Thus, three layers are the best choice for RetailRocket dataset and Diginetica dataset. On the contrary, the average length of sessions in Tmall dataset is relatively long. More extra information excessively supplemented would inevitably obscure the true intention of the current session. Thus, one-layer setting is the best on Tmall dataset.

In summary, the above results tell us that too little information of related sessions makes insufficient learning of the current session, while too much brings some noise. Thus, this experiment proves that, in addition to the current session's information, appropriate supplementation of cross-session information can effectively improve the recommending performance.

Impact of representational dimension. We also study the impact of representational dimension which ranges in {16, 32, 64, 128, 256}. The results are reported in Fig. 6. It can be seen that the performance of HyperS^{2}Rec firstly improves and then decreases slightly with the increment of dimensions. The reason is that a small representational dimension leads to insufficient representational ability, while a large probably leads to over-fitting. Moreover, HyperS^{2}Rec has different optimal representational dimensions on different datasets. The best performance is achieved when the representational dimension is set to 128 on Tmall dataset and Diginetica dataset. Nevertheless, 64 is the best on RetailRocket dataset.

5.8. Complexity comparison

We chose SR-GNN which is designed based on simple graph, SHARE and DHCN which are designed based on hypergraph to test the complexity of the proposed HyperS^{2}Rec. For fairness, we conducted experiments on the same GPU server and recorded their resource consumption, respectively. Table 4 presents the resource consumption on three datasets.

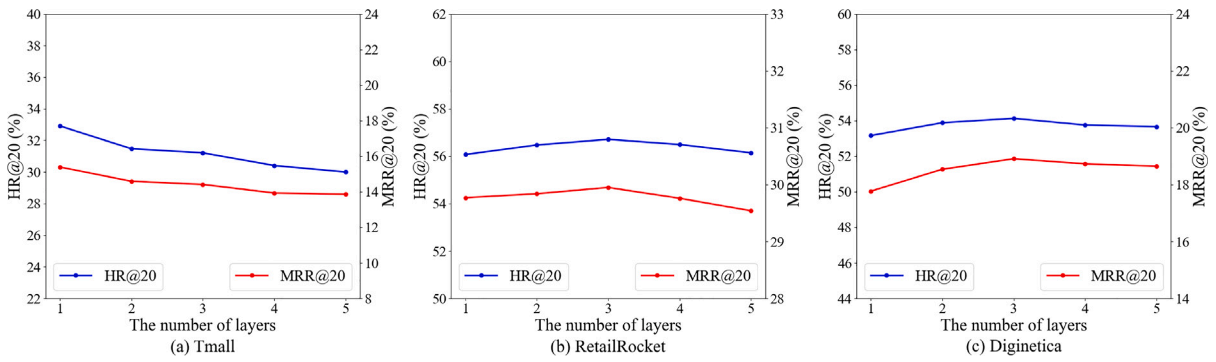

Fig. 5. The performance of HyperS^{2}Rec on different hypergraph convolutional layers.

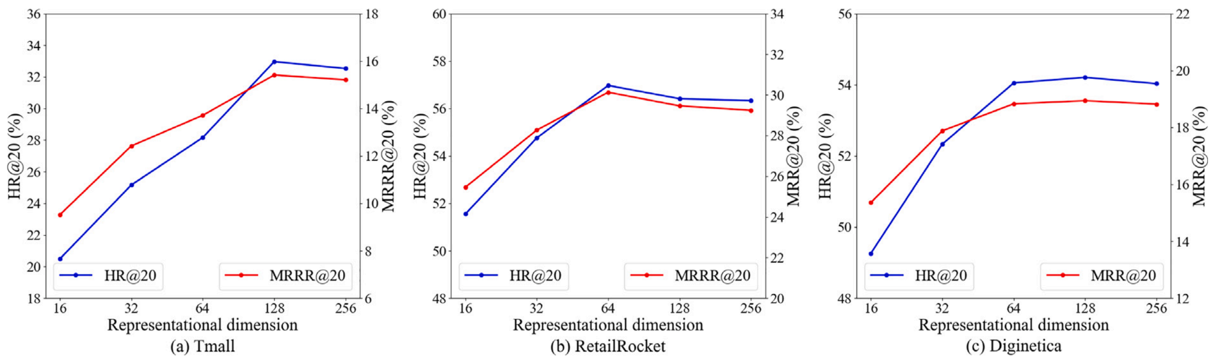

Fig. 6. The performance of HyperS^{2}Rec on different representational dimensions.

Table 4
Space and time consumption of the proposed HyperS^{2}Rec and three baselines over three datasets.

Methods	Tmall			RetailRocket			Diginetica		
	Mem.	GPU Mem.	Time	Mem.	GPU Mem.	Time	Mem.	GPU Mem.	Time
SR-GNN	4813MB	2219MB	248s	5018MB	2695MB	895s	5222MB	2321MB	402s
SHARE	5939MB	2397MB	286s	6554MB	3567MB	1107s	5939MB	2887MB	461s
DHCN	5278MB	2371MB	579s	6349MB	2795MB	1563s	5427MB	2531MB	1147s
HyperS^{2}Rec	5171MB	2324MB	312s	6298MB	2706MB	1204s	5338MB	2355MB	844s

As shown in Table 4, SHARE, DHCN and HyperS^{2}Rec consume more space and time than that of SR-GNN. The main reason is that the former models are designed based on hypergraphs, while the latter is based on simple graph. Particularly, SR-GNN has the smallest space and time consumption on three datasets. Although SHARE and DHCN consume more memory and training time, they both improve the recommending performance to a certain extent. Due to the calculation of attention coefficients and the stacking of HGAT layers, SHARE has relatively more space and time consumption. In addition, DHCN constructs two kinds of graphs, which inevitably bring more space and time consumption than HyperS^{2}Rec. From Table 4, we can see that although the space and time consumed by HyperS^{2}Rec are not the least, they both increase within a reasonable range and it is capable of achieving the best performance which has been proved in Table 3.

6. Conclusion

In this paper, we propose a novel method HyperS^{2}Rec for session-based recommendation. It considers both the consistency and sequential dependence between items, simultaneously. Hypergraph-structured information captured by HGCN and sequential information captured by GRU are exploited to jointly model the preferences of users. The reversed position embedding mechanism and soft attention mechanism are combined to obtain session representations. Experimental results on three real-world datasets demonstrate that the proposed HyperS^{2}Rec is superior to several competitive methods.

Our proposed HyperS^{2}Rec is designed based on clicking behavior, which ignores the impact of other behaviors users generated, such as search, add to favorites and add to cart et al. Such information may reflect users' real preferences from more aspects, and is helpful to predict their intentions accurately. In future work, we will attempt to explore methods that integrate multiple types of behaviors to improve the performance of session-based recommendation.

CRediT authorship contribution statement

Chengxin Ding: Investigation, Writing – original draft, Data curation. **Zhongying Zhao:** Methodology, Supervision, Writing – review & editing, Funding acquisition. **Chao Li:** Conceptualization, Writing – review & editing. **Yanwei Yu:** Writing – review & editing. **Qingtian Zeng:** Methodology, 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

I have share the codes in the abstract via a link.

Acknowledgments

This research is supported by the National Key R&D Program of China (Grant No. 2022ZD0119500), National Natural Science Foundation of China (Grant No. 62072288, 61702306, 61433012), the Taishan Scholar Program of Shandong Province, Shandong Youth Innovation Team, the Natural Science Foundation of Shandong Province, China (Grant No. ZR2018BF013, ZR2022MF268), the Open Project from CAS Key Lab of Network Data Science and Technology (Grant No. CASNDST202007).

References

Aggarwal, C. C. (2016). Content-based recommender systems. In *Recommender systems* (pp. 139–166).

Bai, S., Zhang, F., & Torr, P. H. (2021). Hypergraph convolution and hypergraph attention. *Pattern Recognition*, *110*, Article 107637.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, *12*(4), 331–370.

Chatzis, S. P., Christodoulou, P., & Andreou, A. S. (2017). Recurrent latent variable networks for session-based recommendation. In *Proceedings of the 2nd workshop on deep learning for recommender systems* (pp. 38–45).

Feng, L., Cai, Y., Wei, E., & Li, J. (2022). Graph neural networks with global noise filtering for session-based recommendation. *Neurocomputing*, *472*, 113–123.

Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y. M., & Yuan, Q. (2015). Personalized ranking metric embedding for next new POI recommendation. In *24th International joint conference on artificial intelligence* (pp. 2069–2075).

Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., et al. (2021). Graph neural networks for recommender systems: Challenges, methods, and directions. arXiv preprint arXiv:2109.12843.

Guo, J., Yang, Y., Song, X., Zhang, Y., Wang, Y., Bai, J., et al. (2022). Learning multi-granularity consecutive user intent unit for session-based recommendation. In *Proceedings of the 15th ACM International conference on web search and data mining* (pp. 343–352).

Guo, L., Yin, H., Wang, Q., Chen, T., Zhou, A., & Quoc Viet Hung, N. (2019). Streaming session-based recommendation. In *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 1569–1577).

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th international conference on learning representations* (pp. 1–10).

Hidasi, B., Quadrana, M., Karatzoglou, A., & Tikk, D. (2016). Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on recommender systems* (pp. 241–248).

Jannach, D., & Ludewig, M. (2017). When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the 11th ACM Conference on recommender systems* (pp. 306–310).

Latifi, S., Mauro, N., & Jannach, D. (2021). Session-aware recommendation: A surprising quest for the state-of-the-art. *Information Sciences*, *573*, 291–315.

Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., & Ma, J. (2017). Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on information and knowledge management* (pp. 1419–1428).

Liang, D., Altosaar, J., Charlin, L., & Blei, D. M. (2016). Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM Conference on recommender systems* (pp. 59–66).

Liu, Q., Zeng, Y., Mokhosi, R., & Zhang, H. (2018). STAMP: Short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 1831–1839).

Luo, A., Zhao, P., Liu, Y., Zhuang, F., Wang, D., Xu, J., et al. (2020). Collaborative self-attention network for session-based recommendation. In *Proceedings of the 29th International joint conference on artificial intelligence* (pp. 2591–2597).

Malik, S., Rana, A., & Bansal, M. (2020). A survey of recommendation systems. *Information Resources Management Journal*, *33*(4), 53–73.

Pan, Z., Cai, F., Chen, W., Chen, H., & de Rijke, M. (2020). Star graph neural networks for session-based recommendation. In *Proceedings of the 29th ACM International conference on information and knowledge management* (pp. 1195–1204).

Qiu, R., Huang, Z., Chen, T., & Yin, H. (2021). Exploiting positional information for session-based recommendation. *ACM Transactions on Information Systems*, *40*(2), 1–24.

Qiu, R., Li, J., Huang, Z., & Yin, H. (2019). Rethinking the item order in session-based recommendation with graph neural networks. In *Proceedings of the 28th ACM International conference on information and knowledge management* (pp. 579–588).

Qiu, R., Yin, H., Huang, Z., & Chen, T. (2020). GAG: Global attributed graph neural network for streaming session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on research and development in information retrieval* (pp. 669–678).

Quadrana, M., Karatzoglou, A., Hidasi, B., & Cremonesi, P. (2017). Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the 11th ACM Conference on recommender systems* (pp. 130–137).

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).

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).

Song, B., Cao, Y., Zhang, W., & Xu, C. (2019). Session-based recommendation with hierarchical memory networks. In *Proceedings of the 28th ACM International conference on information and knowledge management* (pp. 2181–2184).

Tan, Y. K., Xu, X., & Liu, Y. (2016). Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on deep learning for recommender systems* (pp. 17–22).

Tang, J., & Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th 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 11th ACM Conference on recommender systems* (pp. 138–146).

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, R., Lou, J., & Jiang, Y. (2022). Session-based recommendation with time-aware neural attention network. *Expert Systems with Applications*, *210*, Article 118395.

Wang, M., Ren, P., Mei, L., Chen, Z., Ma, J., & de Rijke, M. (2019). A collaborative session-based recommendation approach with parallel memory modules. In *Proceedings of the 42nd International ACM SIGIR Conference on research and development in information retrieval* (pp. 345–354).

Wang, Z., Wei, W., Cong, G., Li, X.-L., Mao, X.-L., & Qiu, M. (2020). Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on research and development in information retrieval* (pp. 169–178).

Wu, X., Liu, Q., Chen, E., He, L., Lv, J., Cao, C., et al. (2013). Personalized next-song recommendation in online karaokes. In *Proceedings of the 7th ACM Conference on recommender systems* (pp. 137–140).

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning* (pp. 6861–6871).

Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., & Tan, T. (2019). Session-based recommendation with graph neural networks. In *Proceedings of the 33rd AAAI Conference on artificial intelligence* (pp. 346–353).

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 35th AAAI Conference on artificial intelligence* (pp. 4503–4511).

Xu, C., Zhao, P., Liu, Y., Sheng, V. S., Xu, J., Zhuang, F., et al. (2019). Graph contextualized self-attention network for session-based recommendation. In *Proceedings of the 28th International joint conference on artificial intelligence* (pp. 3940–3946).

You, J., Wang, Y., Pal, A., Eksombatchai, P., Rosenburg, C., & Leskovec, J. (2019). Hierarchical temporal convolutional networks for dynamic recommender systems. In *Proceedings of the 28th International conference on world wide web* (pp. 2236–2246).

Yu, F., Liu, Q., Wu, S., Wang, L., & Tan, T. (2016). A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on research and development in information retrieval* (pp. 729–732).

Yu, F., Zhu, Y., Liu, Q., Wu, S., Wang, L., & Tan, T. (2020). TAGNN: Target attentive graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on research and development in information retrieval* (pp. 1921–1924).

Yue, G., Xiao, R., Zhao, Z., & Li, C. (2022). AF-GCN: Attribute-fusing graph convolution network for recommendation. *IEEE Transactions on Big Data*.

Zhang, X., Lin, H., Xu, B., Li, C., Lin, Y., Liu, H., et al. (2022). Dynamic intent-aware iterative denoising network for session-based recommendation. *Information Processing and Management*, *59*(3), Article 102936.

Zhang, M., Wu, S., Gao, M., Jiang, X., Xu, K., & Wang, L. (2020). Personalized graph neural networks with attention mechanism for session-aware recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *32*, 1–12.

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

Zhang, C., Zheng, W., Liu, Q., Nie, J., & Zhang, H. (2022). SEDGN: Sequence enhanced denoising graph neural network for session-based recommendation. *Expert Systems with Applications*, *203*, Article 117391.

Zhao, Z., Yang, Y., Li, C., & Nie, L. (2021). GuessUNeed: Recommending courses via neural attention network and course prerequisite relation embeddings. *ACM Transactions on Multimedia Computing, Communications, and Applications*, *16*(4), 132:1–132:17.

Zhao, Z., Yang, Z., Li, C., Zeng, Q., Guan, W., & Zhou, M. (2022). Dual feature interaction-based graph convolutional network. *IEEE Transactions on Knowledge and Data Engineering*.

Zhao, Z., Zhang, X., Zhou, H., Li, C., Gong, M., & Wang, Y. (2020). HetNERec: Heterogeneous network embedding based recommendation. *Knowledge-Based Systems*, *204*, Article 106218.

Zheng, Y., Liu, S., Li, Z., & Wu, S. (2020). DGTN: Dual-channel graph transition network for session-based recommendation. In *2020 International conference on data mining workshops* (pp. 236–242).